A robust number theoretic transform with applications in error control and communication systems

Yanto, Jakop

2008

Yanto, J. (2008). A robust number theoretic transform with applications in error control and communication systems. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2552>

<https://doi.org/10.32657/10356/2552>

Nanyang Technological University

Downloaded on 26 Apr 2023 19:35:17 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**A ROBUST NUMBER THEORETIC TRANSFORM
WITH APPLICATIONS IN ERROR CONTROL
AND COMMUNICATION SYSTEMS**

YANTO JAKOP

SCHOOL OF COMPUTER ENGINEERING

2008

A Robust Number Theoretic Transform with Applications in Error Control and Communication Systems

Yanto Jakop

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2008

Abstract

This thesis presents a novel modification for a number theoretic transform (NTT) called *Robust Symmetrical Number System* (RSNS) and addresses its applications in error control and communication systems.

NTTs have very attractive properties, such as fault-tolerant features as well as a lower complexity in computer arithmetic. RSNS is one subclass of NTT that decomposes an integer into a set of parallel residues. Due to the carry free arithmetic and lack of ordered significance among the residue digits, operations to the residues can be carried out in parallel.

RSNS has inherent features, such as short dynamic range and integer Gray property. Due to the short dynamic range, the difference between the representable range and the efficient information dynamic range is significant. This allows self-detection of errors without the need of additional residues as in Residue Number System (RNS).

However, due to the integer Gray property of RSNS, high correlation exists between residue vectors of two consecutive integers. This results in a low error detection probability. To improve the error detection ability, several binary representations, such as binary, Gray and inverse Gray codes are studied for mapping the residues in the context of RSNS. Theoretical and numerical results show that RSNS coded with inverse Gray, referred to as *inverse Gray RSNS* (IGRSNS), outperforms binary and Gray RSNS and has a near-optimal error detection ability.

IGRSNS is further studied for application in error correction. One redundant modulus is added to improve the error correction ability of IGRSNS. An efficient error correction algorithm is proposed. Studies show that IGRSNS with one redundant modulus can improve the error correction ability substantially compared to binary and Gray RSNS.

The good error detection and correction capabilities of IGRSNS make us look more into the potential to be applied in communication systems. A transceiver structure for IGRSNS-CDMA is proposed. The BER performance is evaluated against RNS-CDMA with and without redundancy. Simulation studies show that IGRSNS-CDMA can improve the coding gain of RNS-CDMA by 3-6 dB, and this makes it a good candidate for next generation of broadband communication systems.

Acknowledgements

The entirety of this research is dedicated to the glory of God. May it be used for His purposes and help in some small way to bring peace and wisdom to the world in His name.

I sincerely express my utmost thanks and gratitude towards my supervisor, Associate Professor A. S. Madhukumar, for his invaluable guidance throughout these two years. I would not have reached the current stage and accomplished the project without his endless support, patience and encouragement in the pursuit of my masters degree.

I would also like to thank Associate Professor A. B. Premkumar for his help and constructive ideas in the area of number theoretic transforms, which helped to improve some of my works. I also thank him for reviewing my works and giving advice during the write-up of this thesis.

I wish to thank all my friends and colleagues at Nanyang Technological University, particularly in the Center for Multimedia and Network Technology (CeMNet), who have made my everyday research and teaching life cheerful and enjoyable. In particular, I would like to thank Koh Su Ming, Tio Surya Dharma, Ernest Kurniawan, Sudhan Majhi and Rully Adrian Santosa for their sharing and discussions during the entire period of my masters degree. I would also like to express my gratitude to the laboratory executives of CeMNet, Madam Chua Poo Hua and Miss Oh Hwee May, for their kind assistance in both hardware and software throughout the research.

Finally, I would like to express my greatest gratitude and love to my family, for their care, love and support; especially to my Mom and Dad, who are always there for me whenever I need them most.

Contents

Abstract	ii
Acknowledgements	iv
Table of Contents	v
List of Figures	ix
List of Tables	xi
List of Abbreviations	xiii
List of Publications	xiv
1 Introduction	1
1.1 Background	1
1.2 Motivation	2
1.3 Original Contributions	3
1.4 Organization of Thesis	4
2 Number Theoretic Transforms (NTT)	6
2.1 Some Terminologies in Number Theory	7
2.2 Types of Number Theoretic Transforms	8
2.2.1 Residue Number System (RNS)	9

2.2.2	Redundant Residue Number System (RRNS)	13
2.2.3	Symmetrical Number System (SNS)	14
2.2.4	Optimum Symmetrical Number System (OSNS)	17
2.2.5	Robust Symmetrical Number System (RSNS)	19
2.2.6	Example	20
2.3	Advantages of Number Theoretic Transforms	21
2.4	Applications of Number Theoretic Transforms	22
2.4.1	In Error Control	22
2.4.2	In Communication Systems	26
2.5	Summary	30
3	Robust Symmetrical Number System (RSNS)	32
3.1	Encoding of RSNS	33
3.2	Dynamic Range of RSNS	36
3.3	Decoding of RSNS	37
3.4	Comparison between RSNS and RNS	45
3.5	Implementation of RSNS	48
3.5.1	Implementation Issues in RSNS	49
3.5.2	RSNS Encoding and Decoding Mechanism using LUT	51
3.6	Summary	54
4	Application of RSNS for Error Control	55
4.1	Error Detection in RSNS	55
4.1.1	Definition of Holes in RSNS	55
4.1.2	Analysis of Error Detection Probability in RSNS	56
4.1.3	Integer Gray Code property of RSNS	58
4.2	Binary Representation for RSNS: (1) Binary Code	61
4.2.1	Encoding and Decoding of Binary Code	61
4.2.2	Distance Properties of Binary-coded Integers	62
4.2.3	Error Detection using Binary RSNS	63
4.3	Binary Representation for RSNS: (2) Gray Code	66
4.3.1	Encoding and Decoding of Gray Code	67
4.3.2	Distance Properties of Gray-coded Integers	68

4.3.3	Error Detection using Gray RSNS	68
4.3.4	Suitability of Binary and Gray Code for RSNS	71
4.4	Binary Representation for RSNS: (3) Inverse Gray Code	72
4.4.1	Encoding of Inverse Gray Code	74
4.4.2	Decoding of Inverse Gray Code	76
4.4.3	Distance Properties of Inverse Gray-coded Integers	77
4.4.4	Error Detection using Inverse Gray RSNS	81
4.4.5	Numerical Results: Error Detection Probability of Binary, Gray and Inverse Gray RSNS	83
4.5	Error Correction in RSNS	85
4.5.1	Error Correction without Redundancy	85
4.5.2	Introduction of Redundancy in RSNS	86
4.5.3	Error Correction Algorithm	87
4.5.4	Numerical Results: Average Error Correction Probability of Binary, Gray and Inverse Gray RSNS	94
4.6	Advantages of Inverse Gray RSNS (IGRSNS) over RNS	96
4.7	Summary	97
5	Application of IGRSNS in Digital Communication Systems	98
5.1	Transceiver Structure of IGRSNS-CDMA	99
5.1.1	Pre-transmission: Construction of LUT	99
5.1.2	Transmitter Structure	99
5.1.3	Receiver Structure	103
5.2	Simulation Results	105
5.2.1	Performance Comparison of IGRSNS-CDMA and RNS- CDMA	106
5.2.2	Effect of Moduli Set Selection on BER Performance	110
5.2.3	Performance of IGRSNS-CDMA in Multipath Channel . . .	111
5.3	Summary	112
6	Conclusions and Future Works	113
6.1	Conclusions	113
6.2	Future Works	115

Appendix	119
A Error Detection Ability of Binary, Gray and Inverse Gray Codes	119
B Error Correction Ability of Binary, Gray and Inverse Gray Codes	123
Bibliography	123

List of Figures

2.1	SNS and RNS waveforms for modulus 11	15
2.2	Block diagram of RNS-CDMA transmitter	26
2.3	Block diagram of RNS-CDMA receiver	27
3.1	Residues of a three-moduli RSNS with moduli set $[3\ 4\ 5]$	34
3.2	MRSS plot for moduli set $[3\ 4\ 5]$	38
3.3	Sequence shift in RSNS	50
3.4	Encoding and decoding mechanism in RSNS	51
4.1	Four-bit rotary encoder for Gray and inverse Gray code	74
4.2	Encoding of inverse Gray codes	78
4.3	Decoding of inverse Gray codes	78
4.4	Probability of detection in RSNS for $k = 6$	84
4.5	Error correction algorithm in RSNS	89
4.6	Probability of correction in RSNS for $k = 6$	95
5.1	Block diagram of IGRSNS-CDMA transmitter	100
5.2	(a) Constellation for 16-QAM modulation scheme (b) Constellation for 64-QAM modulation scheme. Integer value in the parenthesis shows the equivalent integer h of the corresponding binary value in inverse Gray representation.	102
5.3	(a) Generalized block diagram of IGRSNS-CDMA Receiver (b) Details of subreceiver for i^{th} residue channel in Figure 5.3(a). (h is the channel information and P is the number of paths)	104
5.4	Modified transceiver structure of RNS-CDMA (The subreceiver at the receiver is similar to the one in Figure 5.3(b).)	107

5.5	Performance comparison of IGRSNS-CDMA and RNS-CDMA (without redundancy)	108
5.6	Performance comparison of IGRSNS-CDMA and RNS-CDMA (with one redundant modulus)	109
5.7	Effect of moduli sets on BER performance	110
5.8	BER performance of IGRSNS-CDMA under AWGN and Rayleigh channels	112
A.1	Probability of Detection in RSNS for $k = 4$	121
A.2	Probability of Detection in RSNS for $k = 5$	121
A.3	Probability of Detection in RSNS for $k = 6$	122
B.1	Probability of Correction in RSNS for $k = 4$	125
B.2	Probability of Correction in RSNS for $k = 5$	125
B.3	Probability of Correction in RSNS for $k = 6$	126

List of Tables

2.1	Residue patterns in RNS with moduli set $\mathbf{m} = [3\ 4\ 5]$	9
2.2	Dynamic range comparison showing advantage of an even modulus	17
2.3	Decimal-to-OSNS mapping	19
2.4	Integer representation with moduli set $[3\ 4\ 5]$	21
3.1	Modulus residue sequence for a three-moduli RSNS	33
3.2	Three-moduli RSNS with moduli set $[3\ 4\ 5]$	34
3.3	A single MRS decimated into three MRSSs	37
3.4	1 st MRSS for moduli set $[3\ 4\ 5]$	39
3.5	2 nd MRSS for moduli set $[3\ 4\ 5]$	39
3.6	3 th MRSS for moduli set $[3\ 4\ 5]$	39
3.7	Two redundancy solutions for residue vector $\mathbf{r} = [0\ 4\ 4]$	44
3.8	Eight redundancy solutions for residue vector $\mathbf{r} = [1\ 3\ 1]$	45
3.9	Comparison of M and $\hat{M}$ for several moduli sets	48
3.10	Encoding of RSNS using Look-up Table (LUT)	52
4.1	Number of holes in RSNS	58
4.2	Three-moduli RSNS with moduli set $\mathbf{m} = [3\ 4\ 5]$	59
4.3	Four-bit binary and Gray codes	61
4.4	Hamming distance d of four-bit binary code	62
4.5	Neighbors of four-bit binary code	63
4.6	Binary RSNS — Single bit error in <i>one</i> MRS for $\mathbf{r} = [2\ 2\ 3]$	64
4.7	Binary RSNS — Single bit error in <i>two</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$	65
4.8	Binary RSNS — Single bit error in <i>all</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$	66
4.9	Hamming distance d of four-bit Gray code	68
4.10	Neighbors of four-bit Gray code	69

4.11	Gray RSNS — Single bit error in <i>one</i> MRS for $\mathbf{r} = [2\ 2\ 3]$	69
4.12	Gray RSNS — Single bit error in <i>two</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$	70
4.13	Gray RSNS — Single bit error in <i>all</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$	71
4.14	Inverse Gray code as derived from Gray code	73
4.15	Three-bit inverse Gray code	74
4.16	3-bit, 4-bit and 5-bit inverse Gray codes	79
4.17	Hamming distance d of four-bit inverse Gray code	80
4.18	Neighbors of four-bit inverse Gray code	80
4.19	Inverse Gray RSNS — Single bit error in <i>one</i> MRS for $\mathbf{r} = [2\ 2\ 3]$.	81
4.20	Inverse Gray RSNS — Single bit error in <i>two</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$.	82
4.21	Inverse Gray RSNS — Single bit error in <i>all</i> MRSs for $\mathbf{r} = [2\ 2\ 3]$.	82
4.22	LUT structure for moduli set $\mathbf{m} = [3\ 4\ 5]$ with checking residue r_4 and $k = 4$	87
4.23	Correction for $\mathbf{r}' = [6\ 2\ 3]$	89
4.24	Correction for $\mathbf{r}' = [0\ 2\ 3]$: Step 1	90
4.25	Correction for $\mathbf{r}' = [0\ 2\ 3]$: Step 2	90
4.26	Correction for $\mathbf{r}' = [2\ 0\ 7]$	91
4.27	Correction for $\mathbf{r}' = [2\ 3\ 2]$: Step 1	92
4.28	Correction for $\mathbf{r}' = [2\ 3\ 2]$: Step 2	92
4.29	Correction for $\mathbf{r}' = [2\ 3\ 2]$: Step 3	92
4.30	Correction for $\mathbf{r}' = [0\ 0\ 7]$: Step 1	93
4.31	Correction for $\mathbf{r}' = [0\ 0\ 7]$: Step 2	93
5.1	Recommended moduli sets and modulation types for several values of bits per symbol b	101
A.1	Probability of error detection P_D in RSNS using binary, Gray and inverse Gray (IG) codes	120
B.1	Probability of error correction P_C in RSNS using binary, Gray and inverse Gray (IG) codes	124

List of Abbreviations

AGC	Anti Gray Coding
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CDMA	Code Division Multiple Access
CRT	Chinese Remainder Theorem
DR	Dynamic Range
GCD	Greatest Common Divisor
IGRSNS	Inverse Gray Robust Symmetrical Number System
ISI	Inter Symbol Interference
LCM	Least Common Multiple
LUT	Look Up Table
MAI	Multiple Access Interference
MRS	Modulus Residue Sequence
MRSS	Modulus Residue Sub-Sequence
NTT	Number Theoretic Transform
OSNS	Optimum Symmetrical Number System
PRP	Pairwise Relatively Prime
QAM	Quadrature Amplitude Modulation
RNS	Residue Number System
RRNS	Redundant Residue Number System
RSNS	Robust Symmetrical Number System
RST	Ratio Statistic Test
SF	Spreading Factor
SNS	Symmetrical Number System

List of Publications

1. **Title:** Inverse Gray RSNS: A New Number Theoretic Transform with Applications in Error Control
Author: Yanto Jakop, A. S. Madhukumar and A. B. Premkumar
Submitted to: IEEE Transactions on Circuits and Systems I
Status: Under review

2. **Title:** A Robust Symmetrical Number System based Parallel Communication System with Inherent Error Detection and Correction
Author: Yanto Jakop, A. S. Madhukumar and A. B. Premkumar
Submitted to: IEEE Transactions on Wireless Communications
Status: Under review

3. **Title:** A Novel Communication System based on Robust Symmetrical Number System
Author: Yanto Jakop, A. S. Madhukumar and A. B. Premkumar
Submitted to: IEEE Communication Letters
Status: Under review

Chapter 1

Introduction

1.1 Background

In recent years, there has been an increasing demand for efficient and reliable data communication. This demand has been accelerated by the emergence of large-scale, high-speed data networks as well as fault-tolerant and ubiquitous computing. The design of these systems requires a merging of communications and computer technology. One major issue in the design of a reliable communication is to control the errors so that reliable reproduction of data can be obtained at the receiver after transmission over some noisy channels [1].

There are many obstacles for a reliable communication system. Some examples are errors in channel estimation, presence of noise, synchronization errors and interference from other users [2]. To deal with some of these obstacles, efficient algorithms for error detection and correction are required.

A class of error detection and correction code is based on *number theoretic transform* (NTT). NTT is basically a technique to transform numbers based on number theory. Usually this transform is achieved by the use of modulo arithmetic. NTT has some very attractive features, such as carry free arithmetic and lack

of ordered significance among residue digits [3] [4]. One typical example of NTT is the Residue Number System (RNS), which has been widely employed in error detection and correction in digital processors, arithmetic units and data transmission. The self-checking architectures were first introduced by Watson and Hastings in [5], which was then developed by Barsi and Maestrini in [6] who introduced the concept of legitimate and illegitimate range, which gave birth to the concept of redundant RNS (RRNS).

RRNS has been widely studied in the literature in the context of fault-tolerant signal processing and error control. It is well known that RRNS is able detect and correct errors, and its detection and correction ability is proportional to the number of redundant moduli. This restriction poses a certain limitation on its performance — that is, additional moduli have to be introduced to achieve a certain correction ability. For example, for correction of single bit errors, at least two redundant moduli are required [3]. These additional moduli translate to a lower code rate, which may not be desirable in some practical applications.

1.2 Motivation

Recently, a new class of number system, named as Robust Symmetrical Number System (RSNS), was proposed by Phillip Pace in [7]. The theory of RSNS was further explored and developed by Luke in his dissertation [8]. Similar to RNS, this number system is also based on number theoretic transform.

RSNS has a lot of attractive features. The main advantage of RSNS over RNS or RRNS is that it has inherent symmetry in its basic sequence, and thus ambiguity is present *without* additional moduli. Besides, the integer Gray coding property makes it particularly attractive for elimination of encoding errors. Due to its robustness, it has been proposed for use in many engineering applications, including folding

analog-to-digital converters [9], direction-finding interferometer antenna architectures [10] and electro-optic digital antennas [11].

The potential of this number theoretic system has not been fully explored. At present, the study of RSNS is still confined to limited application areas in electronic engineering, and the study of coding properties of RSNS is not available in the literature. Besides, the use of RSNS for fault tolerance and error control has not been fully explored. It is in this context that the present research explores the error correction properties of RSNS and addresses the use of RSNS for *robust* and *reliable* data transmission for practical applications.

1.3 Original Contributions

As discussed in the previous section, the inherent fault-tolerant features of RSNS make it very attractive for error control without any additional moduli. However, its performance for error control is far from optimal if conventional binary representation is used with RSNS. This thesis proposes a novel binary representation for residues, named as *Inverse Gray codes*, to improve the performance of RSNS. Extensive theoretical analysis and simulation results have shown that inverse Gray RSNS (IGRSNS) is able to achieve a near-optimal error detection probability without any additional moduli. Besides, it is also able to correct more than 90% of single bit errors with *one* additional modulus.

These high error detection and correction abilities immediately promise two attractive applications of RSNS, namely fault tolerant digital arithmetic as well as digital/wireless communications. In the present research, the proposed scheme is applied for communication systems, and the bit error rate (BER) performance of the resulting system is measured under additive white Gaussian noise (AWGN) and realistic multipath (Rayleigh) channel. Simulation results verify that IGRSNS-

CDMA performs better compared to RNS-CDMA in terms of computational complexity as well as BER performance.

1.4 Organization of Thesis

This thesis is organized as follows:

Chapter 2 provides a literature review on the fundamentals of Number Theoretic Transforms (NTTs). Several types of NTT are briefly introduced, such as the Residue Number System (RNS), Redundant Residue Number System (RRNS), Symmetrical Number System (SNS), Optimal Symmetrical Number System (OSNS) and Robust Symmetrical Number System (RSNS). Several advantages of NTTs are discussed. The applications of NTTs in the field of error control and communication systems that are available in the literature are presented.

Chapter 3 presents the fundamentals and features of Robust Symmetrical Number System (RSNS). The encoding and decoding as well as dynamic range of RSNS are presented. The inherent characteristics of RSNS sequence are compared with RNS, and their implications on efficient implementation of encoding and decoding in RSNS are discussed.

Chapter 4 proposes the application of RSNS for error detection and correction. As opposed to RNS, RSNS by itself can be used for error detection *without* any redundancy. This attractive feature is explored further for use in error control. Several binary mapping methods are introduced for RSNS, including binary, Gray as well as inverse Gray code. *Inverse Gray* code is the novel contribution of this thesis and is applied for RSNS. The encoding and decoding of inverse Gray to/from binary code are presented. The proposed algorithms for error detection and correction using RSNS are also presented. This chapter and the appendices provide computer simulation results based on exhaustive iterations for several moduli sets to justify

the performance comparison of these coding techniques.

Chapter 5 extends the application of RSNS into communication systems. A Code Division Multiple Access (CDMA) communication system based on *inverse Gray RSNS* (IGRSNS) is proposed. Simulation studies are conducted to evaluate the error performance of IGRSNS-CDMA against RNS-CDMA. Besides, the effect of moduli set selection on BER performance is also evaluated. The performance of IGRSNS-CDMA under AWGN and multipath channels is discussed.

Finally, in Chapter 6, the results of the previous chapters are summarized and possible areas for future research are discussed.

Chapter 2

Number Theoretic Transforms (NTT)

In simple terms, a *number theoretic transform* (NTT) is a coding method to transform an integer from one domain into a *set* of integers in another domain. One popular method of transform is achieved by the use of modulo arithmetic. By doing the transform, characteristics of the numbers that are not clearly visible in the original domain may become prominent in a different domain.

Advantages of number theoretic transforms include fault-tolerant properties as well as lower complexity in computer arithmetic due to the use of simple modulo arithmetic, thus leading to an efficient implementation in hardware. NTTs have been widely used in digital arithmetic, digital signal processing and communications. Several examples include the use of NTT for fast convolutions [12] [13] [14], multi-precision multiplications [15], matrix multiplications [16] [17], Cyclic Redundancy Check (CRC) codes [18] [19] [20] and RNS-based Code Division Multiple Access (CDMA) [21] [22] [23].

2.1 Some Terminologies in Number Theory

To make this review and further discussions more understandable, some terminologies pertaining to general number systems are first introduced in this section. Other specialized notations and terminologies will be defined in the body of the report as and when needed.

Modulo arithmetic (residue operation)

The residue of X modulo m , denoted as $|X|_m$ or $X \pmod{m}$ is the least positive remainder when a positive integer X is divided by another positive integer m called *modulus*, i.e.

$$|X|_m = X - m \left\lfloor \frac{X}{m} \right\rfloor \quad (2.1)$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation.

Consequently [5],

$$0 \leq |X|_m < m \quad (2.2)$$

Congruence relationship

If an integer m exactly divides the difference $(a - b)$, we say that $a \equiv |b|_m$ or $|a|_m = |b|_m$. In other words, both a and b have the same residues when divided by m [3].

Example: $5 \equiv |20|_3$, since $|5|_3 = |20|_3 = 2$.

Dynamic range

Dynamic range of a number system is defined as the longest representable sequence of *unique* vectors (i.e. without redundancy or repetition) the number system can provide. The higher the dynamic range, the more the unique vectors the system can represent.

Pairwise relatively prime (PRP)

Integers $x_1, x_2, \dots, x_N$ are said to be *pairwise relatively prime (PRP)* if and only if they do not have a common integer divisor of absolute value greater than 1, i.e. their greatest common divisor (gcd) is 1. In other words, $\gcd(x_i, x_j) = 1$ for all $i \neq j$ and $0 < i, j \leq N$. Note that each integer x_i need not be a prime number itself.

Example: Integers $\{6, 7, 9\}$ are not PRP since $\gcd(6, 9) = 3$, while integers $\{5, 7, 9\}$ are PRP since $\gcd(5, 7) = \gcd(7, 9) = \gcd(5, 9) = 1$.

2.2 Types of Number Theoretic Transforms

Residue-based number theoretic transforms can be classified into two large subgroups: the *residue number system (RNS)* and *symmetrical number system (SNS)*. Both RNS and SNS operate based on modulo arithmetic; however, SNS employs an inherent symmetry and ambiguity in its basic sequence whereas RNS uses a straightforward modulo operation. Variants of RNS include redundant RNS (RRNS), which incorporate ambiguity by adding redundant residues. Variants of SNS include optimum SNS (OSNS) and robust SNS (RSNS). A brief description of each transform is provided in the following sections.

2.2.1 Residue Number System (RNS)

A residue number system (RNS) is a number theoretic transform that represents a large integer using a set of smaller integers, so that computation on the integers may be performed more efficiently. This number system relies on the use of modulo arithmetic for its operation, a mathematical idea inspired by a Chinese scholar, Sun Tsu Suan-Ching in the 4th century [24].

Encoding of RNS

RNS is defined by a set of N positive integers $[m_1 \ m_2 \ \dots \ m_N]$ referred to as *moduli*. Letting M to be the least common multiple (lcm) of all moduli m_i , any arbitrary positive integer X smaller than M can be represented in the number system defined as a set of N smaller integers $[r_1 \ r_2 \ \dots \ r_N]$ called *residues* with $r_i = |X|_{m_i}$, i.e.

$$X \Leftrightarrow (r_i)_{i=1}^N = (|X|_{m_i})_{i=1}^N \quad (2.3)$$

where the double-headed arrow $\Leftrightarrow$ means "corresponds to" and $| \cdot |_{m_i}$ represents a modulo operation with respect to modulus m_i [5].

For maximum representational efficiency, it is imperative that all the moduli values are pairwise relatively prime (PRP); that is, no modulus may have a common factor with any other moduli [3].

Example: Integers in a three-moduli residue number system with moduli set $\mathbf{m} = [3 \ 4 \ 5]$ have the following residue sequences:

Integer X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
$r_1 = X _3$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	...
$r_2 = X _4$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	...
$r_3 = X _5$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	...

Table 2.1: Residue patterns in RNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$

Decoding of RNS

In RNS, decoding is achieved by the use of Chinese Remainder Theorem (CRT). According to the CRT, given a residue sequence $[r_1 \ r_2 \ \dots \ r_N]$ where $0 \leq r_i < m_i$ and $i = 1, 2, \dots, N$, the corresponding integer X in the range $[0, M)$ can be uniquely recovered from the residue digits using [25]:

$$X = \left| \sum_{i=1}^N r_i T_i M_i \right|_M, \quad (2.4)$$

where $M_i = \frac{M}{m_i}$ and the integers T_i that constitute the multiplicative inverses of M_i , are computed apriori by solving the congruence equation:

$$T_i M_i \equiv |1|_{m_i} \quad (2.5)$$

Equation (2.4) can also be expressed as

$$X = \left| \sum_{i=1}^N r'_i M_i \right|_M, \quad (2.6)$$

where r'_i is computed by

$$r'_i M_i \equiv |r_i|_{m_i} \quad (2.7)$$

Example: Consider an RNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. A residue vector $\mathbf{r} = [2 \ 0 \ 3]$ corresponds to integer X . This integer X can be determined by using

the congruence equations in equation (2.7) as follows:

$$\begin{aligned} r'_1 \left(\frac{60}{3} \right) &\equiv |2|_3 \\ r'_2 \left(\frac{60}{4} \right) &\equiv |0|_4 \\ r'_3 \left(\frac{60}{5} \right) &\equiv |3|_5 \end{aligned}$$

Solving these congruence equations, we obtain $r'_1 = 1$, $r'_2 = 0$ and $r'_3 = 4$. Substituting these values to equation (2.6),

$$X = |1(20) + 0(15) + 4(12)|_{60} = |68|_{60} = 8$$

Besides CRT, another popular algorithm to carry out the RNS decoding is the mixed radix conversion approach [26].

Dynamic Range of RNS

Dynamic range of RNS with moduli set $[m_1 \ m_2 \ \dots \ m_N]$, denoted as M , is the least common multiple (lcm) of all moduli m_i , i.e.

$$M = \text{lcm}(m_1, m_2, \dots, m_N) \quad (2.8)$$

Recall from number theory that lcm of a set of numbers $\{m_1, m_2, \dots, m_N\}$ is the smallest non-zero number that is a common multiple of all these numbers. Given that each number m_i has prime factors $\{b_1^{e_{i1}}, b_2^{e_{i2}}, \dots, b_P^{e_{iP}}\}$, where $\{b_1, b_2, \dots, b_P\}$ denotes the set of unique *base* prime factors, $\{e_{i1}, e_{i2}, \dots, e_{iP}\}$ denotes the *exponent* corresponding to each prime factor, and P is the number of unique prime

factors of these N numbers, the lcm equation is given by

$$\text{lcm}(m_1, m_2, \dots, m_N) = \prod_{i=1}^P b_i^{\max(e_{1i}, e_{2i}, \dots, e_{Ni})} \quad (2.9)$$

Thus, if all moduli values in the moduli set $[m_1 \ m_2 \ \dots \ m_N]$ are chosen to be PRP, the dynamic range of RNS M is the product of all moduli [5], i.e.

$$M = \text{lcm}(m_1, m_2, \dots, m_N) = \prod_{i=1}^N m_i \quad (2.10)$$

Consequently, since $\max(\text{lcm}(m_1, m_2, \dots, m_N)) = \prod_{i=1}^N m_i$, the dynamic range of RNS is *optimal* when the moduli are chosen to be PRP.

Example: An RNS with non-PRP moduli set $\mathbf{m} = [6 \ 7 \ 9]$ has a dynamic range of $M = \text{lcm}(6, 7, 9) = 126 \not\geq 6 \times 7 \times 9$. On the other hand, an RNS with PRP moduli set $\mathbf{m} = [5 \ 7 \ 9]$ has a dynamic range of $M = \text{lcm}(5, 7, 9) = 5 \times 7 \times 9 = 315$.

Within this dynamic range M , any addition, subtraction or multiplication of two integers represented by residue sequences preserve the correspondence in equation (2.3), as long as modulo operation is performed and all the operands fall in the range of 0 to $M - 1$; in symbols, [5]

$$X * Y \leftrightarrow \left(|X|_{m_i} * |Y|_{m_i} \right)_{i=1}^N \quad (2.11)$$

where symbol $*$ represents arithmetic operation such as ordinary addition, subtraction or multiplication of two integers.

This makes RNS very attractive, since any operation on the individual residues is independent of each other and the original number can be constructed back from these residues provided all operations are carried out in the Galois field of each moduli value.

2.2.2 Redundant Residue Number System (RRNS)

A redundant residue number system (RRNS) is obtained by appending additional r number of redundant moduli $m_{N+1}, m_{N+2}, \dots, m_{N+R}$ to the previously introduced RNS, forming an RRNS code of $N+R$ positive and PRP moduli, i.e. $[m_1 \ m_2 \ \dots \ m_N \ m_{N+1} \ m_{N+2} \ \dots \ m_{N+R}]$ [24] [27] [28]. These additional moduli should obey the condition [23]

$$\{m_{N+1}, \dots, m_{N+R}\} \geq \max\{m_1, \dots, m_N\} \quad (2.12)$$

Now a positive integer X smaller than M is represented as an $N+R$ -tuple residue sequence with respect to the $N+R$ moduli $[m_1 \ m_2 \ \dots \ m_{N+R}]$, expressed as

$$X \Leftrightarrow (r_i)_{i=1}^{N+R} = (|X|_{m_i})_{i=1}^{N+R}, \quad (2.13)$$

which is referred to as an RRNS codeword or RRNS code vector. The residue vector $\mathbf{r}$ of each integer X is divided into two parts, i.e. the information part and the parity part. The information part consists of the first non-redundant N residue digits $r_1, r_2, \dots, r_N$ and the parity part consists of the remaining r redundant parity check digits $r_{N+1}, r_{N+2}, \dots, r_{N+R}$. The parity check digits are residue digits derived from the information residue digits. Correspondingly, the first N moduli are termed as the non-redundant moduli, while the additional R moduli are the redundant moduli. This code is a systematic code as integer X , $0 \leq X < M$, can be uniquely determined from the first N components of the residue vector $\mathbf{r}$.

The total range of RRNS code is therefore

$$MM_R = \prod_{i=1}^{N+R} m_i \quad (2.14)$$

where $M = \prod_{i=1}^N m_i$ is called the *legitimate range* and $M_R = \prod_{i=1}^R m_{N+i}$ is the *illegitimate range* [27]. Since redundancies in RRNS are used for error control, the extra residues are only meant for error checking and probable correction; they are not meant to increase the dynamic range of RNS. Thus, although the range of representable vectors is $M \times M_R$, the *effective* information dynamic range M in equation (2.10) still holds for RRNS.

2.2.3 Symmetrical Number System (SNS)

Similar to RNS and RRNS, Symmetrical Number System (SNS) is based on modulo arithmetic. However, different from RNS and RRNS, SNS provides a symmetrical waveform with respect to each modulus.

The symmetrically folded waveform corresponding to each SNS modulus m_i has a period equal to the modulus. The integer values within each SNS modulus are derived from a mid-level quantization of a symmetrical folding waveform and are in-congruent modulo m_i . For example, the SNS symmetrical waveform associated with $m_i = 5$ is $\{0, 1, 2, 2, 1, 0, 1, 2, \dots\}$. It has a basic waveform of $\{0, 1, 2, 2, 1\}$ and a period of 5. This SNS formulation is somewhat similar to redundant RNS in the sense that it allows ambiguities to occur. The ambiguities that arise within SNS are resolved by using various arrangements of the SNS moduli.

Definition of SNS

Given an integer X such that $0 \leq X < m$, an SNS integer value r_X can be obtained by [9]

$$r_X = \min\{X, m - X\} \quad (2.15)$$

This function can be extended periodically with period m , that is

$$r_{X+nm} = r_X \quad (2.16)$$

where $n \in \{0, \pm 1, \pm 2, \dots\}$

Thus, the SNS vector for a given modulus m is

$$\mathbf{r} = \begin{cases} [0, 1, \dots, \lfloor \frac{m}{2} \rfloor, \lfloor \frac{m}{2} \rfloor, \dots, 2, 1] & \text{if } m \text{ odd;} \\ [0, 1, \dots, \frac{m}{2}, \frac{m}{2} - 1, \dots, 2, 1] & \text{if } m \text{ even.} \end{cases} \quad (2.17)$$

The SNS definition above is illustrated in Figure 2.1 for modulus 11. The SNS folding waveform is symmetrical about the midpoint and smoother as compared with the corresponding RNS sawtooth waveform.

Figure 2.1: SNS and RNS waveforms for modulus 11

Due to the presence of ambiguities, this SNS vector does not form a complete system of length m by themselves [11]. The ambiguities that arise within the modulus are resolved by considering the paired values from all channels together. By recombining N channels, the SNS is rendered as a complete system having a one-to-one correspondence with the RNS. A complete SNS representation of an integer is built with respect to a given PRP moduli set $[m_1 \ m_2 \ \dots \ m_N]$.

Dynamic range of SNS

For PRP moduli set $\mathbf{m}$ with one of the moduli even, the first repetitive vector occurs at position [11]

$$\hat{M} = \min \left\{ \frac{m_1}{2} \prod_{l=2}^j m_{il} + \prod_{l=j+1}^N m_{il} \right\} \quad (2.18)$$

where j ranges from 2 to $N - 1$ and $i_2, i_3, \dots, i_N$ range over all permutations $\{2, 3, \dots, N\}$. This $\hat{M}$ is also the dynamic range of the system.

As an example, an SNS matrix with moduli set $\mathbf{m} = [2 \ 5 \ 7]$ has the first repetitive column vector occurring at $X = \min\{1 \times 5 + 7, 1 \times 7 + 5\} = 12$. Thus, this system can uniquely represent X for $0 \leq X \leq 11$ as shown in Table 2.2.

For PRP moduli set $\mathbf{m}$ with all moduli odd, the first repetitive vector occurs at position [11]

$$\hat{M} = \min \left\{ \frac{1}{2} \prod_{l=1}^j m_{il} + \frac{1}{2} \prod_{l=j+1}^N m_{il} \right\} \quad (2.19)$$

where j ranges from 1 to $N - 1$ and $i_1, i_2, \dots, i_N$ range over all permutations $\{1, 2, \dots, N\}$.

Thus, the SNS with all odd PRP moduli can be put in a one-to-one correspondence with a complete number system over the dynamic range defined by

Integer	SNS Moduli					
	2	5	7	3	5	7
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	0	2	2	1	2	2
3	1	2	3	0	2	3
4	0	1	3	1	1	3
5	1	0	2	1	0	2
6	0	1	1	0	1	1
7	1	2	0	1	2	0
8	0	2	1	1	2	1
9	1	1	2	0	1	2
10	0	0	3	1	0	3
11	1	1	3	1	1	3
12	0	2	2	0	2	2

Table 2.2: Dynamic range comparison showing advantage of an even modulus

equation (2.19). For example, an SNS matrix with moduli set $\mathbf{m} = [3 \ 5 \ 7]$ has the first repetitive column vector occurring at $X = \min\{\frac{3+5 \times 7}{2}, \frac{5+3 \times 7}{2}, \frac{7+3 \times 5}{2}\} = \min\{19, 13, 11\} = 11$. Thus, this system can uniquely represent X for $0 \leq X \leq 10$ and is shown in Table 2.2. For proof of equations (2.18) and (2.19), refer to [29].

It is noted that the *reduction* in the value of modulus m_1 from $m_1 = 3$ (odd modulus) to $m_1 = 2$ (even modulus) results in an *increase* in total dynamic range. Also note that dynamic range of SNS is smaller than that of RNS, i.e. $\hat{M} < M$. For a comparable-sized system, the SNS for moduli set with one even modulus gives a larger dynamic range than an all-odd moduli set, and results in a more efficient scheme.

2.2.4 Optimum Symmetrical Number System (OSNS)

Since the SNS folding waveform is symmetrical, ambiguities exist within each folding period or modulus. Consequently, dynamic range of SNS depends on the SNS definition and the manner in which the row vectors (or channels) are recombined. Optimum SNS (OSNS) can considerably extend the dynamic range of SNS, and it was shown to be optimum. For the complete proof, refer to [30].

Definition of OSNS

The optimum SNS is composed of a number of PRP moduli $[m_1 \ m_2 \ \dots \ m_N]$. The integers within each OSNS modulus m_i are quantized values of a symmetrically folded waveform with the period of *twice* the PRP modulus value (i.e., $2m_i$). For a given modulus m , the OSNS integer values within twice the individual modulus are given by the row vector

$$\mathbf{r} = [0, 1, \dots, m-1, m-1, \dots, 1, 0] \quad (2.20)$$

Similar with SNS, due to the presence of ambiguities, the integers within equation (2.20) do not form a complete system of length $2m$ by themselves [31]. The ambiguities that arise within the modulus are resolved by considering paired values from all channels together. By recombining the N channels together, OSNS is rendered a complete system having one-to-one correspondence with RNS [30].

Dynamic range of OSNS

For N equal to the number of PRP moduli, the dynamic range of OSNS is [32]

$$M = \prod_{i=1}^N m_i \quad (2.21)$$

This dynamic range is also the position of the first repetitive vector. For example, consider an OSNS system with $m_1 = 3$ and $m_2 = 4$, the first repetitive vector occurs at an input of 12 as shown in Table 2.3.

Integer X	OSNS Moduli	
	3	4
0	0	0
1	1	1
2	2	2
3	2	3
4	1	3
5	0	2
6	0	1
7	1	0
8	2	0
9	2	1
10	1	2
11	0	3
12	0	3

Table 2.3: Decimal-to-OSNS mapping

2.2.5 Robust Symmetrical Number System (RSNS)

As seen in the previous section, OSNS has optimum dynamic range. However, similar to RNS and SNS, the transition of OSNS residues from one integer to the next has no obvious pattern.

Robust Symmetrical Number System (RSNS) is an SNS-based number-theoretic transform that contains integer Gray-code properties, i.e. RSNS residues only change by one integer position from one integer to the next.

Definition of RSNS

In the RSNS, N different periodic symmetrical waveforms are used with PRP moduli set $[m_1 \ m_2 \ \dots \ m_N]$. The RSNS is formed based on the following row vector

$$\mathbf{r} = [0, 1, 2, \dots, m-1, m, m-1, \dots, 2, 1] \quad (2.22)$$

Note that unlike in RNS and other SNS systems, the largest integer within each periodic sequence is the modulus m instead of $m-1$.

In an N -channel RSNS where $N \geq 2$, the basic single-modulus residue se-

quence (MRS) for the i th channel (modulus m_i) is

$$\mathbf{r}_i = [0 \cdots 0, 1 \cdots 1, \cdots, m_i \cdots m_i, \cdots, 1 \cdots 1] \quad (2.23)$$

In this sequence, each value of $\mathbf{r}$ in equation (2.22) is used N times in succession, forming a sequence with length $2mN$.

Similar to other SNS systems, due to the presence of ambiguities, the integers within a single MRS in equation (2.22) do not form a complete system of length $2m$ by themselves. The ambiguities that arise within the modulus are resolved by considering paired values from all channels together. However, even after combining N channels together, it can be easily shown that the RSNS vectors are still not unique. To make them unique, different shift values $\{s_1, s_2, \dots, s_N\}$ need to be introduced in each channel. Due to the relative property of the shifts, one of the shift values must be set to 0, and all shift values s_i must be different from one another.

Dynamic Range of RSNS

In general, there are no closed form expression to find the dynamic range of an N -moduli RSNS. The dynamic range has to be computed by using computer search algorithm. Dynamic range of RSNS is shorter compared to OSNS.

2.2.6 Example

The following example is provided to illustrate above number systems. Assuming moduli set $\mathbf{m} = [3 \ 4 \ 5]$ and shift values $\mathbf{s} = [0 \ 1 \ 2]$ for RSNS, integer $0 \leq X \leq 12$ can be represented in RNS, SNS, OSNS and RSNS as shown in Table 2.4.

Integer X	RNS	SNS	OSNS	RSNS
0	0 0 0	0 0 0	0 0 0	0 0 0
1	1 1 1	1 1 1	1 1 1	0 0 1
2	2 2 2	1 2 2	2 2 2	0 1 1
3	0 3 3	0 1 2	2 3 3	1 1 1
4	1 0 4	1 0 1	1 3 4	1 1 2
5	2 1 0	1 1 0	0 2 4	1 2 2
6	0 2 1	0 2 1	0 1 3	2 2 2
7	1 3 2	1 1 2	1 0 2	2 2 3
8	2 0 3	1 0 2	2 0 1	2 3 3
9	0 1 4	0 1 1	2 1 0	3 3 3
10	1 2 0	1 2 0	1 2 0	3 3 4
11	2 3 1	1 1 1	0 3 1	3 4 4
12	0 0 2	0 0 2	0 3 2	2 4 4

Table 2.4: Integer representation with moduli set [3 4 5]

2.3 Advantages of Number Theoretic Transforms

From the above descriptions, there are several advantages that can be observed regarding number theoretic transforms. Firstly, after the transform, the residues are much smaller than the original number. A large integer can be transformed into smaller residues, provided small moduli are used and dynamic range of these moduli is larger than the integer itself. This makes NTT ideally suitable for application in computer hardware and electronics, where in most cases, the use of fewer number of bits is desired. Fewer number of bits means lower computational complexity, and thus reduction in processing time as well as cost.

Number theoretic transforms have several important properties, such as [3] [4]:

- Carry free arithmetic

This implies that any operation on residues (addition, subtraction and multiplication) related to different residue digits are mutually independent and can be carried out simultaneously in parallel. Besides, any errors during arithmetic operation or due to transmission noise remains confined to their original residues. In other words, these errors do not propagate and hence do not contaminate other residue digits due to the absence of a carry forward.

- Lack of ordered significance among residue digits

This implies that any erroneous digit can be discarded without affecting the result, provided that enough redundancy is added.

These properties make number theoretic transforms attractive for implementation into a wide range of practical applications, some of which includes digital processing, fault-tolerant computing, error control and communication systems.

2.4 Applications of Number Theoretic Transforms

Number theoretic transforms have been widely used in many engineering applications. RNS and RRNS, for example, have been used for fast convolutions [12] [13] [14], multi-precision multiplications [15], matrix multiplications [16] [17], Cyclic Redundancy Check (CRC) codes [18] [19] [20] and Code Division Multiple Access (CDMA) systems [21] [22] [23]. SNS, OSNS and RSNS have been used for under-sampling digital receivers, electro-optical digital antennas, folding analog to digital converter (ADC) and direction-finding antennas [11].

A survey on practical applications of NTT has been provided by Gudvangen in [15]. In the following section, the application of one subclass of NTT, namely the Residue Number System (RNS), in the area of error control and communication systems is presented.

2.4.1 In Error Control

Error detection and/or correction is required for the protection of arithmetic computation and data transmission in digital processors as well as in general purpose computers. An RNS by itself has no fault-tolerant properties; detection and correction of errors are achieved by the use of redundant moduli.

Error Detection

Error detection in RRNS can be achieved by several methods. An intuitive but computationally impractical method is to compare the received vector with every possible codevector in the RRNS dynamic range M . If the received vector does not exist within M , error is detected. Otherwise, either correct reception or undetectable errors are said to have occurred. Another method is by using Chinese Remainder Theorem (CRT) to decode the non-redundant residues $r_1, r_2, \dots, r_N$ into integer X . The result of modulo arithmetic of X with respect to the redundant moduli $m_{N+1}, m_{N+2}, \dots, m_{N+R}$ is compared with the redundant residues $r_{N+1}, r_{N+2}, \dots, r_{N+R}$. If any of these values does not match, error is detected. To illustrate this method, an example is provided as follows.

Example: Consider an RRNS with moduli set $\mathbf{m} = [3 \ 4 \ 5 \ 7]$ where 3, 4, 5 are the non-redundant moduli and 7 is the redundant modulus. The information dynamic range is $M = 3 \times 4 \times 5 = 60$, hence valid residues are defined in the range of $[0, 60)$. Consider an integer $X = 21$ is transmitted, the corresponding residue vector is $\mathbf{r} = [0 \ 1 \ 1 \ 0]$. If there is an error in the RNS representation due to transmission or processing, for example r_3 is changed from 1 to 3, then the received RNS vector becomes $\mathbf{r}' = [0 \ 1 \ 3 \ 0]$. Following the CRT decoding algorithm in Section 2.2.1, using the first three residue digits and their moduli, we obtain

$$M_1 = \frac{60}{3} = 20, M_2 = \frac{60}{4} = 15, M_3 = \frac{60}{5} = 12,$$

$$T_1 = 2, T_2 = 3, T_3 = 3,$$

$$X = |0(2)(20) + 1(3)(15) + 3(3)(12)|_{60} = |153|_{60} = 33$$

However, because $|X|_{m_4} = |33|_7 = 5 \neq r_4 = 0$, error is detected in the RNS representation. Therefore, upon designing RNS with *one* redundant modulus, error

in single residue digit r_3 can be detected.

Error Correction

Similar with detection, error correction in RRNS can also be achieved by several methods. An intuitive but slow method is to use maximum likelihood decoding (MLD), in which the received residue vector in error $\mathbf{r}'$ is compared with all residue vectors in the dynamic range M , and the residue vector having the smallest Hamming distance with $\mathbf{r}'$ is determined to be the corrected vector $\hat{\mathbf{r}}$. Another method, which is more practical, is by considering all possible combinations of residue digits in $\mathbf{r}'$ and decoding each of them individually. The integer that occurs most frequently within the dynamic range M is considered to be the corrected value. To illustrate this method, an example is provided as follows.

Example: Now, consider an RRNS with the same moduli set as the previous example with an additional redundant modulus, namely $m_5 = 11$, resulting in a moduli set of $\mathbf{m} = [3 \ 4 \ 5 \ 7 \ 11]$. Consider the same integer $X = 21$ is transmitted, the corresponding residue vector is $\mathbf{r} = [0 \ 1 \ 1 \ 0 \ 10]$. Assume that r_3 is in error and it was changed from 1 to 3, i.e. the received vector is $\mathbf{r}' = [0 \ 1 \ 3 \ 0 \ 10]$. According to CRT, integer X in the range $[0, 60)$ can be recovered by invoking any three moduli and their corresponding residue digits, if no errors occurred in the received RNS representation. Integer X can be recovered from $\mathbf{r}' = [0 \ 1 \ 3 \ 0 \ 10]$ by considering

all possible combinations of three out of five residue digits as follows:

$$\begin{aligned}
(r_1, r_2, r_3) = (0, 1, 3) &\Leftrightarrow X_{123} = |33|_{60}, \\
(r_1, r_2, r_4) = (0, 1, 0) &\Leftrightarrow X_{124} = |21|_{84}, \\
(r_1, r_2, r_5) = (0, 1, 10) &\Leftrightarrow X_{125} = |21|_{132}, \\
(r_1, r_3, r_4) = (0, 3, 0) &\Leftrightarrow X_{134} = |63|_{105}, \\
(r_1, r_3, r_5) = (0, 3, 10) &\Leftrightarrow X_{135} = |153|_{165}, \\
(r_1, r_4, r_5) = (0, 0, 10) &\Leftrightarrow X_{145} = |21|_{231}, \\
(r_2, r_3, r_4) = (1, 3, 0) &\Leftrightarrow X_{234} = |133|_{140}, \\
(r_2, r_3, r_5) = (1, 3, 10) &\Leftrightarrow X_{235} = |153|_{220}, \\
(r_2, r_4, r_5) = (1, 0, 10) &\Leftrightarrow X_{245} = |21|_{308}, \\
(r_3, r_4, r_5) = (3, 0, 10) &\Leftrightarrow X_{345} = |98|_{385},
\end{aligned}$$

where X_{ijk} represents the recovered result by using moduli m_i , m_j and m_k as well as their corresponding residue digits r_i , r_j and r_k . From these results, it is observed that X_{134} , X_{135} , X_{234} , X_{235} and X_{345} are all illegitimate numbers, since their values are out of the legitimate range $[0, 60)$. In the remaining five cases, except for X_{123} , all the results are the same and equal to 21. Moreover, all these results were recovered from three moduli without including m_3 , i.e. from X_{124} , X_{125} , X_{145} and X_{245} , which are equal to 21. Hence, it is concluded that the correct result is 21 and that there was an error in r_3 , which can be corrected by computing $\hat{r}_3 = |21|_5 = 1$.

The above examples give a deduction on the error detecting and correcting capability of RRNS: If there are R redundant moduli, it is possible to detect R errors and correct $\lfloor \frac{R}{2} \rfloor$ errors. In other words, error correction in RRNS can be achieved only if more than one redundant modulus is present.

2.4.2 In Communication Systems

Besides error control, NTT is also widely applied in the context of communication systems. For example, the application of RNS and RRNS in Code Division Multiple Access (CDMA) systems has been widely discussed in the literature.

RNS-CDMA was originally proposed by Yang and Hanzo in [21] [22] [33] [34] and later modified by Madhukumar and Chin in [23]. Before RNS-CDMA was introduced, a widely known method to reduce multiple access interference (MAI) was to use M -ary orthogonal CDMA systems [35], whose complexity increases *exponentially* with the number of bits per symbol. In contrast, RNS-CDMA can achieve a receiver complexity which increases *linearly* with the number of bits per symbol.

In RNS-CDMA, any bit string with length b representing information symbol X can be uniquely and unambiguously represented as a set of residues $[r_1 \ r_2 \ \dots \ r_N]$ with respect to a set of PRP moduli $[m_1 \ m_2 \ \dots \ m_N]$, where r_i represents the remainder of X with respect to m_i for $i = 1 \dots N$ and $m_1 < m_2 < \dots < m_N$. The dynamic range is equivalent to the product of all moduli, i.e. $M = \prod_{i=1}^N m_i$ and determines the maximum number of bits possible in a symbol b , where $b \leq \lfloor \log_2 M \rfloor$.

Transmitter Structure

The RNS-CDMA transmitter structure proposed by Yang and Hanzo (assuming three-moduli RNS) is shown in Figure 2.2 [34].

Figure 2.2: Block diagram of RNS-CDMA transmitter

At the transmitter, an M -ary symbol X is encoded into its RNS representation $[r_1 \ r_2 \ r_3]$ by using modulo arithmetic. Each residue r_i is then used for selection of spread code $U_i(r_i)$. Since each residue r_i is within the range of $\{0, 1, 2, \dots, m_i - 1\}$, the number of unique spread codes needed for the RNS-CDMA system is the sum of all moduli, i.e. $\sum_{i=1}^N m_i$. The selected spread codes $U_i(r_i)$ from each residue r_i are then combined together and transmitted.

Receiver Structure

The RNS-CDMA receiver structure proposed by Yang and Hanzo (assuming three-moduli RNS) is shown in Figure 2.3 [34].

Figure 2.3: Block diagram of RNS-CDMA receiver

At the receiver, the noise-perturbed signal is independently correlated with the same set of orthogonal spread sequences $U_i(r'_i)$ for each residue r'_i respectively. The RNS residues $\hat{r}_i$ are determined by maximum likelihood decoding (MLD) from the correlator outputs. These residues are then decoded using inverse RNS transform based on *Chinese Remainder Theorem (CRT)* to obtain the original symbol transmitted X' [34].

Advantages of RNS-CDMA

There are several advantages of RNS-CDMA compared to conventional M -ary orthogonal CDMA systems:

- Mutually independent residues

The operations related to the individual residue channels are mutually independent due to the carry-free nature of residue arithmetic. This leads to the design of a parallel system by decomposing the overall receiver into a set of parallel, independent residue receivers. Also, some residue integers can be discarded without affecting the result, provided that a sufficiently high number of residue integers are retained for the reconstruction of the symbol.

- Fewer number of spread codes

In RNS-CDMA, the number of unique spread codes needed for the RNS-CDMA system is the sum of all moduli, i.e. $\sum_{i=1}^N m_i$. This is a significant improvement over conventional M -ary orthogonal CDMA systems whose number of spread codes is equal to the range of M -ary symbol X , which is the product of all moduli $\prod_{i=1}^N m_i$.

As a comparison, consider an RNS-CDMA with moduli set $\mathbf{m} = [3\ 4\ 5\ 7\ 11]$. Dynamic range for RNS is given by $M = \prod_{i=1}^5 m_i = 4620$, and the maximum symbol length is $b = \lfloor \log_2 M \rfloor = 12$. In RNS-CDMA, the number of unique orthogonal spread codes required is $\sum_{i=1}^5 m_i = 30$. So, to transmit 12-bit symbol, only 5 residues are required, and five orthogonal waveforms corresponding to these five residues can be chosen from a set of 30 orthogonal signals. On the other hand, in the conventional M -ary orthogonal CDMA systems, the number of orthogonal signals required will be $2^{12} = 4096$, several orders of magnitude larger than that of the RNS-CDMA system.

- Lower receiver complexity

In contrast with M -ary orthogonal CDMA systems whose receiver complexity increases linearly with M and exponentially with the number of bits per symbol b , receiver complexity of RNS-CDMA systems is almost independent

of M and b . They are not mutually exclusive, however, because as number of bits per symbol b increases, the dynamic range required increases and thus a higher moduli set is needed.

Since dynamic range of RNS is equal to the product of all moduli, a slight increase in the value of the modulus gives rise to a large increase in the dynamic range. For example, dynamic range of a 3-modulus RNS with moduli set $\mathbf{m}_1 = [m-1 \ m \ m+1]$ (m even) is $M_1 = m^3 - m$. If now the moduli set becomes $\mathbf{m}_2 = [m+1 \ m+2 \ m+3]$, dynamic range is now $M_2 = m^3 + 6m^2 + 11m + 6$. Dynamic range of these two moduli sets differs by $\Delta M = M_2 - M_1 = 6m^2 + 12m + 6$ (order of 2). In general, dynamic range changes on the order of $N - 1$ with the modulus value m for an N -moduli RNS.

Thus, receiver complexity of RNS-CDMA increases at a rate that is *lower than exponential* with the number of bits per symbol b .

- Error detection and correction capability

This can be achieved by the use of redundant moduli as in Redundant RNS (RRNS). Some additional information, such as the *Ratio Statistic Test* (RST), may also be used to improve the error correction capability of RNS-CDMA [36]. RST shows the reliability of the residues, and hence the original information can be retrieved by decoding only the most reliable residues having the highest values of RST.

Limitations of RNS-CDMA

Although RNS-CDMA has very attractive features, there are also some inherent limitations such as:

- Decoder Complexity

Although decoder complexity in RNS-CDMA is generally lower compared to that of M -ary orthogonal CDMA systems, decoding of RNS is non-trivial in practice due to the use of Chinese Remainder Theorem (CRT) which requires the computation of multiplicative inverse. Due to the complexity of solving CRT equations, RNS-CDMA becomes unattractive for implementation compared to other channel codes.

- Poor Error Performance

RNS by itself gives poor error performance due to its inability to detect and correct errors, as all received vectors fall within the legitimate range (i.e. the Galois field). As described previously, this problem can be alleviated with RRNS. However, the correction ability depends on the number of redundant moduli. If better correction ability is desired, more redundant moduli have to be added. This addition of redundant moduli translates to a larger spreading factor and larger bandwidth requirement. Unfortunately, trading off bandwidth efficiency for good error performance may not be desirable in many practical applications.

2.5 Summary

Number theoretic transforms have some attractive advantages, such as error detection and correction ability as well as lower computational complexity. So far, the analysis of number theoretic transform is only restricted to one subclass of number theoretic transform, namely RNS and RRNS. Only few studies have been focused on other types of NTTs, such as SNS, OSNS and RSNS.

RNS and RRNS have not been implemented widely in practice due to its hardware complexity. Besides, there are obvious limitations for specific applications

such as in communications where additional bandwidth is required and error performance is unsatisfactory. It is in this context that this thesis analyzes an alternative number theoretic transform based on *Robust Symmetrical Number System* (RSNS) for applications in error control and communication systems. Further research shows that RSNS has a lot of attractive features that make it potential to replace RNS. Some fundamentals of RSNS are provided in the next chapter.

Chapter 3

Robust Symmetrical Number System (RSNS)

Robust Symmetrical Number System (RSNS) is a number theoretic transform that was recently proposed by Phillip Pace in [7]. Similar to RNS, RSNS decomposes a large integer X into N smaller integer residues. However, different from RNS, these residues are not obtained by modulo arithmetic. Instead, they are obtained by a fixed pre-defined symmetrical periodic sequence. Due to this symmetry, RSNS has a structure with built-in redundancies within the basic sequence itself without adding redundant moduli as in RNS. Besides, the *integer Gray code* properties make this number system very useful in many engineering applications. RSNS has been proposed for use in folding analog-to-digital converters [9], direction-finding interferometer antenna architectures [10] and electro-optic digital antennas [11].

A brief introduction of RSNS has been provided in Chapter 2. This chapter extends the theoretical analysis and discussion of RSNS — the general properties, encoding, decoding and its theoretical comparison against RNS. Some implications of inherent RSNS properties will be discussed, and a more efficient encoding and decoding mechanism for practical implementation is presented.

3.1 Encoding of RSNS

Similar to RNS, RSNS is also defined by a moduli set consisting of N positive integers $[m_1 \ m_2 \ \dots \ m_N]$. However, RSNS *residues*¹ are not obtained by modulo arithmetic. Instead, they are formed by a basic *symmetrical* residue structure $\mathbf{r}_i$ for each modulus m_i .

The structure for a single *modulus residue sequence* (MRS) with modulus m_i in an N -moduli RSNS (where $N \geq 2$) is [8]

$$\mathbf{r}_i = [0 \cdots 0, 1 \cdots 1, \dots, m_i \cdots m_i, \dots, 1 \cdots 1] \quad (3.1)$$

Each integer residue is repeated N times, forming an MRS with length $2m_iN$, called a *modulus period*.

For example, in a three-moduli case, the RSNS single-modulus residue sequence with modulus m_i and its corresponding integer X are shown in Table 3.1. The modulus period of this MRS is $6m_i$.

r_i	0	0	0	1	1	1	$\cdots$	m_i	m_i	m_i	$\cdots$	1	1
X	0	1	2	3	4	5	$\cdots$	$3m_i$	$3m_i + 1$	$3m_i + 2$	$\cdots$	$6m_i - 2$	$6m_i - 1$

Table 3.1: Modulus residue sequence for a three-moduli RSNS

Due to the presence of ambiguities (repeated residue values), the integers within each MRS in equation (3.1) do not form a complete residue system by themselves. These ambiguities are resolved by taking into account the combined values from N MRSs together formed by N moduli in the moduli set $\mathbf{m} = [m_1 \ m_2 \ \dots \ m_N]$ [37].

When forming the RSNS, all moduli $[m_1 \ m_2 \ \dots \ m_N]$ are required to be pairwise relatively prime (PRP). Once the PRP moduli are chosen, each MRS corresponding to m_i ($i = 1, 2, \dots, N$) is formed according to Table 3.1. Next, MRS for

¹In fact, the term "residue" is not very appropriate for RSNS, since no direct modulo arithmetic is applied. But to make it consistent with RNS, this term is still maintained to refer to the integer values within each SNS modulus in the transform domain.

the *second* modulus m_2 is circularly shifted left by *one* position; MRS for the *third* modulus m_3 is circularly shifted left by *two* positions relative to the non-shifted MRS. This process is repeated for all remaining MRSs until the last MRS with modulus m_N , which is circularly shifted left by $N - 1$ positions relative to the non-shifted MRS. Finally, all the N MRSs $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N$ are repeated a number of times and stacked together to form residue vectors $\mathbf{r}$ [8].

As an example, a portion of three-moduli RSNS structure with $\mathbf{m} = [3 \ 4 \ 5]$ is shown in Table 3.2.

$\mathbf{r}$	$r_1(m_1 = 3)$	0	0	0	1	1	1	2	2	2	3	3	3	2	2	2	...
	$r_2(m_2 = 4)$	0	0	1	1	1	2	2	2	3	3	3	4	4	4	3	...
	$r_3(m_3 = 5)$	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	...
X		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Table 3.2: Three-moduli RSNS with moduli set $[3 \ 4 \ 5]$

The *integer Gray code* property can be clearly seen by noticing the transition between residue vector of integer X and $X + 1$ in Table 3.2. All elements of the vectors remain the same except for *one* MRS value and this MRS value varies by ± 1 . The plot of these residues is shown in Figure 3.1 [8].

Figure 3.1: Residues of a three-moduli RSNS with moduli set $[3 \ 4 \ 5]$

Plotting the residues r_i for each MRS versus integer X yields a folded, stair-case structure which clearly shows the result of the left-shift in the second and third MRSs. For consistency, the analysis in this research assumes that the MRS corresponding to moduli m_1 , m_2 and m_3 are always left-shifted circularly by 0, 1 and 2 positions respectively, and $m_1 < m_2 < m_3$. It has been shown that the dynamic range $\hat{M}$ in any particular RSNS is independent of which MRS receives the left-shifts. However, the location of $\hat{M}$ is affected by the choice of MRS that receives the left-shifts [9].

Since the modulus period of one MRS is $6m_i$, the fundamental period of a three-moduli RSNS with PRP moduli set $\mathbf{m} = [m_1 \ m_2 \ m_3]$ is therefore

$$P_f = \text{lcm}(6m_1, 6m_2, 6m_3) \quad (3.2)$$

$$= 6m_1m_2m_3 \quad (3.3)$$

where $\text{lcm}(a, b, c)$ stands for the least common multiple of integers a , b and c .

In general, the residue values of an N -moduli RSNS with moduli $[m_1 \ m_2 \ \dots \ m_N]$ and left-shift values of $[0 \ 1 \ \dots \ N-1]$ can be expressed as

$$r_i = \begin{cases} \lfloor \frac{|X+i-1|_{2Nm_i}}{N} \rfloor, & 0 \leq |X|_{2Nm_i} \leq Nm_i - i \\ 2m_i - \lfloor \frac{|X+i-1|_{2Nm_i}}{N} \rfloor, & Nm_i - i + 1 \leq |X|_{2Nm_i} \leq 2Nm_i - i \end{cases} \quad (3.4)$$

where r_i is the i^{th} residue and $i = 1, 2, \dots, N$, N is the number of moduli, X is the position index (integer value), $|\cdot|$ denotes modulo arithmetic and $\lfloor \cdot \rfloor$ denotes flooring operation.

3.2 Dynamic Range of RSNS

Since RSNS is periodic, unambiguous decoding can only be accomplished within a finite range of vectors. This range of vectors, also called *dynamic range* and denoted as $\hat{M}$, is the largest series of consecutive non-redundant RSNS vectors. For RNS, $\hat{M}$ is simply denoted M and is the product of the moduli. Due to the inherent redundancy (symmetry) in RSNS, the dynamic range $\hat{M}$ is shorter compared to RNS and is very difficult to compute [11]. So far, closed form expression for dynamic range of N -moduli RSNS is not available in the literature.

Computer search algorithms have to be used to find the position and length of $\hat{M}$ in the RSNS fundamental period. Because of the large RSNS fundamental period, this approach becomes cumbersome and slow for systems with many moduli or systems with large moduli. Recently, Luke has developed an efficient computer search algorithm, called *SmartSearch*, to determine the position and length of $\hat{M}$ in [8]. Rather than searching the entire fundamental period for a sequence of non-redundant vectors of *unknown* length, an extensive redundancy analysis is used to compute *finite* locations of the redundancies for each MRS. These redundancy locations are then compared and the longest series of non-redundant RSNS vectors is determined to be the dynamic range $\hat{M}$. For the details of this redundancy analysis, refer to Chapter 4 of his dissertation [8].

Based on the analysis of three-moduli RSNS, an analytical expression for the position and length of $\hat{M}$ is possible for RSNS with moduli set of the form $[m - 1 \ m \ m + 1]$. The start and end position of the longest sequence of this moduli set can be shown to be [8]

$$\begin{aligned} start &= \frac{3}{2}(m^3 - 3m) + 1 \\ end &= \frac{3}{2}(m^3 + m^2) + 1 \end{aligned} \tag{3.5}$$

Consequently, dynamic range $\hat{M}$ is given by [8]

$$\begin{aligned}\hat{M} &= end - start + 1 \\ &= \frac{3}{2}m^2 + \frac{9}{2}m + 1\end{aligned}\tag{3.6}$$

It is very likely that more special classes of moduli exist for the RSNS N -moduli case that have closed form expressions for $\hat{M}$. However, for now, *SmartSearch* can be used to find $\hat{M}$ for an arbitrary set of N moduli.

3.3 Decoding of RSNS

For efficient decoding of RSNS, the formulation of a straightforward analytical relationship between integer X and each integer residue r_i for each modulus m_i is necessary. However, this formulation is very difficult due to the stair-case form of the MRS plot. This can be made simpler by decimating each MRS into three sub-sequences (because $N = 3$ and each integer residue is repeated thrice in three-moduli RSNS) called a *modulus residue sub-sequence* (MRSS). The first MRSS is composed of values from the original MRS at positions where $X \equiv |0|_3$, the second MRSS at $X \equiv |1|_3$ and the third MRSS at $X \equiv |2|_3$.

An example is given at Table 3.3 where the first MRS $\mathbf{r}_1$ from Table 3.2 ($m_1 = 3$) is decimated into three subsequent MRSSs $\mathbf{r}_{11}$, $\mathbf{r}_{12}$ and $\mathbf{r}_{13}$.

$\mathbf{r}_1$	$m_1 = 3$	0	0	0	1	1	1	2	2	2	3	3	3	2	2	2	...
$\mathbf{r}_{11}$	$X \equiv 0 _3$	0			1			2			3			2			...
$\mathbf{r}_{12}$	$X \equiv 1 _3$		0			1			2			3			2		...
$\mathbf{r}_{13}$	$X \equiv 2 _3$			0			1			2			3			2	...
X		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Table 3.3: A single MRS decimated into three MRSSs

The top row in Table 3.3 is the original MRS for $m_1 = 3$ and below it are the

three MRSSs and finally the integer (position index) X . Notice that the positions of the MRSS integers do not align with each other. Every integer value X falls within one and only one MRSS. Figure 3.2 provides a plot of the three MRSSs in each MRS for $\mathbf{m} = [3 \ 4 \ 5]$.

Figure 3.2: MRSS plot for moduli set $[3 \ 4 \ 5]$

The position index X of each MRSS can be aligned and re-indexed to simplify the congruence equations, where the relationship between the old index X and the new index Y is given by

$$Y = \frac{X}{3} \quad (3.7)$$

for the first MRSS,

$$Y = \frac{X - 1}{3} \quad (3.8)$$

for the second MRSS, and

$$Y = \frac{X - 2}{3} \quad (3.9)$$

for the third MRSS. The RSNS for the three re-indexed MRSSs for moduli set $[3 \ 4 \ 5]$ are shown in Table 3.4, Table 3.5 and Table 3.6.

$\mathbf{r}_{11}(m_1 = 3)$	0	1	2	3	2	1	0	1	2	3	2	1	0	1	2	...
$\mathbf{r}_{21}(m_2 = 4)$	0	1	2	3	4	3	2	1	0	1	2	3	4	3	2	...
$\mathbf{r}_{31}(m_3 = 5)$	0	1	2	3	4	5	4	3	2	1	0	1	2	3	4	...
X	0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	...
$Y = \frac{X}{3}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Table 3.4: 1st MRSS for moduli set $[3 \ 4 \ 5]$

$\mathbf{r}_{12}(m_1 = 3)$	0	1	2	3	2	1	0	1	2	3	2	1	0	1	2	...
$\mathbf{r}_{22}(m_2 = 4)$	0	1	2	3	4	3	2	1	0	1	2	3	4	3	2	...
$\mathbf{r}_{32}(m_3 = 5)$	1	2	3	4	5	4	3	2	1	0	1	2	3	4	5	...
X	1	4	7	10	13	16	19	22	25	28	31	34	37	40	43	...
$Y = \frac{X-1}{3}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Table 3.5: 2nd MRSS for moduli set $[3 \ 4 \ 5]$

$\mathbf{r}_{13}(m_1 = 3)$	0	1	2	3	2	1	0	1	2	3	2	1	0	1	2	...
$\mathbf{r}_{23}(m_2 = 4)$	1	2	3	4	3	2	1	0	1	2	3	4	3	2	1	...
$\mathbf{r}_{33}(m_3 = 5)$	1	2	3	4	5	4	3	2	1	0	1	2	3	4	5	...
X	2	5	8	11	14	17	20	23	26	29	32	35	38	41	44	...
$Y = \frac{X-2}{3}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Table 3.6: 3th MRSS for moduli set $[3 \ 4 \ 5]$

Note that each integer residue except 0 and m_i occurs twice in a single MRSS period, with one occurrence in the increasing (folding-up) part of the waveform and the other in the decreasing (folding-down) part of the waveform. Thus, using this fact, the congruence equations describing the position of a residue vector for the 1st

MRSS are

$$\begin{aligned}
 Y &\equiv |r_1|_{2m_1} \quad \text{or} \quad Y \equiv |2m_1 - r_1|_{2m_1} \\
 Y &\equiv |r_2|_{2m_2} \quad \text{or} \quad Y \equiv |2m_2 - r_2|_{2m_2} \\
 Y &\equiv |r_3|_{2m_3} \quad \text{or} \quad Y \equiv |2m_3 - r_3|_{2m_3},
 \end{aligned} \tag{3.10}$$

where the left equations are for the folding-up waveform and right equations are for the folding-down waveform.

Consequently, the congruence equations for the 2nd MRSS are

$$\begin{aligned}
 Y &\equiv |r_1|_{2m_1} \quad \text{or} \quad Y \equiv |2m_1 - r_1|_{2m_1} \\
 Y &\equiv |r_2|_{2m_2} \quad \text{or} \quad Y \equiv |2m_2 - r_2|_{2m_2} \\
 Y &\equiv |r_3 - 1|_{2m_3} \quad \text{or} \quad Y \equiv |2m_3 - r_3 - 1|_{2m_3},
 \end{aligned} \tag{3.11}$$

and congruence equations for the 3rd MRSS are

$$\begin{aligned}
 Y &\equiv |r_1|_{2m_1} \quad \text{or} \quad Y \equiv |2m_1 - r_1|_{2m_1} \\
 Y &\equiv |r_2 - 1|_{2m_2} \quad \text{or} \quad Y \equiv |2m_2 - r_2 - 1|_{2m_2} \\
 Y &\equiv |r_3 - 1|_{2m_3} \quad \text{or} \quad Y \equiv |2m_3 - r_3 - 1|_{2m_3},
 \end{aligned} \tag{3.12}$$

The goal of decoding is, given an RSNS residue vector $\mathbf{r} = [r_1 \ r_2 \ r_3]^T$, we need to find the index Y by solving the systems of congruence equations in equations (3.10), (3.11) and (3.12). The index Y can then be converted to integer X using equations (3.7), (3.8) and (3.9) based on the residue vector $\mathbf{r}$. Because there are three rows of congruence equations for each MRSS with two possible choices in each row in equations (3.10), (3.11) and (3.12), each residue vector can produce up to eight systems of congruence equations.

Note, however, that if a particular MRS residue is a maximum ($r_i = m_i$) or a minimum ($r_i = 0$), the congruence equation *choices* corresponding to the MRS

with the maximum or minimum reduce to a single equation

$$Y \equiv |r_i|_{2m_i} \quad (3.13)$$

or

$$Y \equiv |r_i - 1|_{2m_i} \quad (3.14)$$

depending on the form of the original MRS equation in (3.10), (3.11) or (3.12).

This is because the left equation and the right equation are identical, i.e. $|r_i|_{2m_i} = |2m_i - r_i|_{2m_i}$ and $|r_i - 1|_{2m_i} = |2m_i - r_i - 1|_{2m_i}$.

In order to determine which set of congruence equations (3.10), (3.11) or (3.12) is to be applied to a particular residue vector, it is necessary to scrutinize the residue vectors in Tables 3.4, 3.5 and 3.6. A noticeable even-odd pattern occurs in the residue vectors (even = e , odd = o). In Table 3.4, the residue vectors at even indexes are all even and the residue vectors at odd indexes are all odd. Therefore, the 1st MRSS equations in equation (3.10) are applied to the residue vector when the residues are either all even or all odd, i.e. $[e\ e\ e]^T$ or $[o\ o\ o]^T$. Extending this analysis to the other two MRSSs, the 2nd MRSS equations in equation (3.11) are applied when the r residue pattern is $[e\ e\ o]^T$ or $[o\ o\ e]^T$, and the 3rd MRSS equations in equation (3.12) are applied when the r residue pattern is $[e\ o\ o]^T$ or $[o\ e\ e]^T$.

Note that residue vectors with pattern $[e\ o\ e]^T$ or $[o\ e\ o]^T$ do not exist in the MRSSs shown in this chapter. This is because the first, second and third MRSs are shifted left by zero, one and two positions respectively, while each residue is repeated three times in each MRS. Since the MRS shifts are consecutive and their sum equals the number of residue repetitions, there will always be at least two adjacent residues with the same parity in each three-moduli RSNS vector. If the

RSNS was formed by shifting the MRSs in a different manner, the vector patterns $[e\ o\ e]^T$ or $[o\ e\ o]^T$ could exist and would correspond to a particular MRSS [8].

The discussion so far has resulted in a procedure to find the integer X that corresponds to residues $\mathbf{r} = [r_1\ r_2\ r_3]^T$ within dynamic range $\hat{M}$. The procedure is as follows:

- Select a set of congruence equations from equations (3.10), (3.11) or (3.12) based on the even-odd pattern of the RSNS residue vector
- Expand the congruence equation choices into one, two, four or eight systems of congruence equations depending on the values of r_1 , r_2 and r_3
- Compute the solution to each system of congruence equations using Chinese Remainder Theorem (CRT) to find index Y
- Convert index Y to X using equations (3.7), (3.8) or (3.9) based on the even-odd pattern of the RSNS vector residues
- The unique solution X that falls within $\hat{M}$ is the only position of residue vector $\mathbf{r} = [r_1\ r_2\ r_3]^T$ within $\hat{M}$

Example 3.1

The following example illustrates encoding, decoding as well as calculation of dynamic range $\hat{M}$ for RSNS. Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3\ 4\ 5]$.

Encoding. Assume an integer $X = 108$ is transmitted. The moduli periods for $\mathbf{m} = [3\ 4\ 5]$ are 18, 24 and 30 respectively. Each integer residue is repeated three times in a 3-modulus RSNS and the number of left-shifts in each moduli is 0, 1 and 2 for m_1 , m_2 and m_3 respectively. From equation (3.4), integer 108 corresponds

to $\mathbf{r} = [\lfloor \frac{108+0|_{18}}{3} \rfloor \lfloor \frac{108+1|_{24}}{3} \rfloor \lfloor \frac{108+2|_{30}}{3} \rfloor] = [0 \ 4 \ 6]$. Due to the MRS symmetry, all residues r_i greater than moduli value m_i are folded back to fall within the range $0 \leq r_i \leq m_i$, where $i = 1, 2, 3$, yielding residue vector $\mathbf{r} = [0 \ 4 \ 2(5)-6] = [0 \ 4 \ 4]$.

Dynamic Range. The location of the longest sequence and dynamic range of this number system can be calculated by equations (3.5) and (3.6) respectively. Substituting value of $m = 4$, the longest sequence is in the range $[start, end] = [79, 121]$. The length is therefore $\hat{M} = 43$.

Decoding. Assume that the residue vector received is $\mathbf{r} = [0 \ 4 \ 4]$. The goal is to find the integer value X within $\hat{M}$ that corresponds to the residue vector $\mathbf{r}$. Following the procedure described earlier, the first step is to recognize that the residue vector has the pattern $[e \ e \ e]^T$, which means that this residue vector is from the 1st MRSS. Since the first residue r_1 is minimum ($r_1 = 0$) and the second residue r_2 is maximum ($r_2 = m_2$), applying the congruence equations for 1st MRSS from (3.10), we obtain

$$\begin{aligned} Y &\equiv |0|_6 \\ Y &\equiv |4|_8 \\ Y &\equiv |4|_{10} \text{ or } Y \equiv |6|_{10} \end{aligned} \tag{3.15}$$

The congruence equation choices in (3.15) can be expanded to form two unique systems of congruence equations. The solution to each system of congruence equations can be found using CRT, which yields index Y . Applying the 1st MRSS index conversion $X = 3Y$ from (3.7) to the solutions for index Y yields all positions X for residue vector $\mathbf{r} = [0 \ 4 \ 4]$. The solution that lies within range $[79, 121]$ is the solution of X . Thus $X = 108$.

$Y \equiv 0 _6$		
$Y \equiv 4 _8$	$\rightarrow$	$Y = 84 \quad X = 252$
$Y \equiv 4 _{10}$		
$Y \equiv 0 _6$		
$Y \equiv 4 _8$	$\rightarrow$	$Y = 36 \quad X = 108$
$Y \equiv 6 _{10}$		

Table 3.7: Two redundancy solutions for residue vector $\mathbf{r} = [0 \ 4 \ 4]$

Example 3.2

Consider an RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. Now, assume an integer $X = 87$ is transmitted.

Encoding. By equation (3.4), $X = 87$ corresponds to $\mathbf{r} = [\lfloor \frac{87+0|_{18}}{3} \rfloor \ \lfloor \frac{87+1|_{24}}{3} \rfloor \ \lfloor \frac{87+2|_{30}}{3} \rfloor] = [5 \ 5 \ 9]$. Next, due to the MRS symmetry, all residue(s) r_i greater than moduli value m_i are folded back to fall within the range $0 \leq r_i \leq m_i$, where $i = 1, 2, 3$, yielding $\mathbf{r} = [2(3)-5 \ 2(4)-5 \ 2(5)-9] = [1 \ 3 \ 1]$.

Decoding. Assume now that the residue vector $\mathbf{r} = [1 \ 3 \ 1]$ needs to be decoded. The goal is to find the integer value X within $\hat{M}$ that corresponds to the residue vector $\mathbf{r}$. This residue vector has the pattern $[o \ o \ o]^T$, which means that it is from the 1st MRSS. Since none of the values contain any maximum or minimum values, applying the 1st MRSS congruence equations from 3.10 yields

$$\begin{aligned}
 Y &\equiv |1|_6 \quad \text{or} \quad Y \equiv |5|_6 \\
 Y &\equiv |3|_8 \quad \text{or} \quad Y \equiv |5|_8 \\
 Y &\equiv |1|_{10} \quad \text{or} \quad Y \equiv |9|_{10}
 \end{aligned} \tag{3.16}$$

The congruence equation choices in (3.16) can be expanded to form eight unique systems of congruence equations. The solution to each system of congruence equations can be found using CRT, yielding index Y . Similar to Example 3.1, the 1st MRSS index conversion $X = 3Y$ from equation (3.7) to the solutions for index Y yields all positions X for residue vector $\mathbf{r} = [1 \ 3 \ 1]$. The solution that lies within

range $[79, 121]$ is the solution of X . Thus $X = 87$.

$Y \equiv 1 _6$ $Y \equiv 3 _8$ $Y \equiv 1 _{10}$	$\rightarrow$	$Y = 91$	$X = 273$
$Y \equiv 1 _6$ $Y \equiv 3 _8$ $Y \equiv 9 _{10}$	$\rightarrow$	$Y = 19$	$X = 57$
$Y \equiv 1 _6$ $Y \equiv 5 _8$ $Y \equiv 1 _{10}$	$\rightarrow$	$Y = 61$	$X = 183$
$Y \equiv 1 _6$ $Y \equiv 5 _8$ $Y \equiv 9 _{10}$	$\rightarrow$	$Y = 109$	$X = 327$
$Y \equiv 5 _6$ $Y \equiv 3 _8$ $Y \equiv 1 _{10}$	$\rightarrow$	$Y = 11$	$X = 33$
$Y \equiv 5 _6$ $Y \equiv 3 _8$ $Y \equiv 9 _{10}$	$\rightarrow$	$Y = 59$	$X = 177$
$Y \equiv 5 _6$ $Y \equiv 5 _8$ $Y \equiv 1 _{10}$	$\rightarrow$	$Y = 101$	$X = 303$
$Y \equiv 5 _6$ $Y \equiv 5 _8$ $Y \equiv 9 _{10}$	$\rightarrow$	$Y = 29$	$X = \mathbf{87}$

Table 3.8: Eight redundancy solutions for residue vector $\mathbf{r} = [1 \ 3 \ 1]$

3.4 Comparison between RSNS and RNS

Encoder Complexity

Encoding in both RNS and RSNS is straightforward. In RNS, residues can be computed by modulo arithmetic, while in RSNS, residues can be computed by equation (3.4). These simple arithmetic operations make the encoder complexity of RNS and RSNS relatively low.

Decoder Complexity

In contrast, decoding in RNS and RSNS makes use of *Chinese Remainder Theorem* (CRT). This technique uses multiplicative inverse which requires a relatively complex hardware to compute. In RNS, one CRT computation is needed to decode each residue vector. In RSNS, the number of CRTs needed varies depending on the value of the residues. In a three-moduli case, for example, the number of CRTs needed varies between 1, 2, 4 or 8 depending on the number of congruence equations formed in equations (3.10), (3.11) or (3.12).

Given an RSNS moduli set $\mathbf{m} = [m_1 \ m_2 \ m_3]$, the probability of having minimum or maximum residues (i.e. $r_i = 0$ or $r_i = m_i$) in each modulus m_i (assuming uniform residue probability) is

$$P(r_i = 0 \cup r_i = m_i) = \frac{2}{m_i + 1} \quad (3.17)$$

Hence, on average, the number of CRT computation needed is

$$\begin{aligned} & \left[\prod_{i=1}^3 \left(\frac{2}{m_i+1} \right) \right] \times 1 + \left[\sum_{i=1}^3 \left(1 - \frac{2}{m_i+1} \right) \prod_{\substack{j=1 \\ j \neq i}}^3 \left(\frac{2}{m_j+1} \right) \right] \times 2 + \\ & \left[\sum_{i=1}^3 \left(\frac{2}{m_i+1} \right) \prod_{\substack{j=1 \\ j \neq i}}^3 \left(1 - \frac{2}{m_j+1} \right) \right] \times 4 + \left[\prod_{i=1}^3 \left(1 - \frac{2}{m_i+1} \right) \right] \times 8 \end{aligned} \quad (3.18)$$

As value of m_i increases, the probability of having maximum and minimum residues (i.e. $r_i = m_i$ or $r_i = 0$) in equation (3.17) decreases, hence the average number of CRT computation needed in equation (3.18) approaches 8. This implies that in the worst case, the RSNS decoder complexity is about 8 times higher compared to RNS for a 3-moduli RSNS. In general, for an N -moduli RSNS, the decoder complexity can increase up to 2^N times of that of an equivalent N -moduli RNS. This computational complexity makes it difficult to implement in practice,

and suggests that a more efficient decoding method for RSNS needs to be explored.

Dynamic Range

Due to the inherent redundancy of RSNS sequence, dynamic range of RSNS $\hat{M}$ is significantly shorter compared to RNS. Since there is no closed form expression available for N -moduli RSNS, the dynamic range comparison between RSNS and RNS can only be evaluated by extensive computer simulations using arbitrary moduli sets.

For comparison, consider a three-moduli system with moduli set of the form $\mathbf{m} = [m-1 \ m \ m+1]$. From equation (3.6), an RSNS with this moduli set has a dynamic range $\hat{M} = \frac{3}{2}m^2 + \frac{9}{2}m + 1$. In contrast, an RNS with this moduli set has a dynamic range $M = \prod_{i=1}^N m_i = m^3 - m$. Since function M has an order of $\mathcal{O}(m^3)$ and $\hat{M}$ has an order of $\mathcal{O}(m^2)$, intuitively the dynamic range of RNS is larger than that of RSNS. It can however be shown that the dynamic range of RNS is greater. Defining δ to be the gap (difference) between M and $\hat{M}$, we have

$$\begin{aligned} \delta &= M - \hat{M} \\ &= m^3 - \frac{3}{2}m^2 - \frac{11}{2}m - 1 \end{aligned} \tag{3.19}$$

Taking the first and second derivatives of the gap equation, we have

$$\begin{aligned} \delta' &= 3m^2 - 3m - \frac{11}{2} \\ \delta'' &= 6m - 3 \end{aligned} \tag{3.20}$$

Since gradient δ' and second derivative δ'' are always positive for $m \geq 4$ (the smallest value of m required for a PRP moduli set of the form $[m-1 \ m \ m+1]$ is 4), the gap δ is a convex function, i.e. it is monotonically increasing. Therefore, it can be concluded that $\hat{M}$ is always smaller than M for any moduli values.

Dynamic range of RNS and RSNS for several PRP moduli sets of the form $[m-1 \ m \ m+1]$ is summarized in Table 3.9.

Moduli Set	M	$\hat{M}$	start	end
[61 62 63]	238266	6046	357214	363259
[51 52 53]	140556	4291	210679	214969
[41 42 43]	74046	2836	110944	113779
[29 30 31]	26970	1486	40336	41851
[23 24 25]	13800	973	20629	21601
[17 18 19]	5814	568	8668	9235
[13 14 15]	2730	358	4054	4411
[9 10 11]	990	196	1456	1651
[7 8 9]	504	133	733	865
[3 4 5]	60	43	79	121

Table 3.9: Comparison of M and $\hat{M}$ for several moduli sets

Position of the Longest Sequence

In RNS, the longest sequence is located in $[0, M)$, where M is the dynamic range. In RSNS, due to the symmetry and redundancy within each MRS, the position of longest residue sequence $\hat{M}$ in RSNS does not start from 0, but from the *start* to *end* position obtained from equation (3.5).

3.5 Implementation of RSNS

As mentioned in the previous section, there are some basic distinctions between RSNS and RNS, such as the dynamic range, decoding complexity, and position of the longest sequence. These differences pose additional issues and considerations in the practical implementation of RSNS, such as the architectural design of RSNS encoder and decoder. Some of these implementation issues are discussed in the following section.

For analysis, in this research we assume the use of three-moduli RSNS with moduli set of the form $[m-1 \ m \ m+1]$. The result of this research can be extended

to a general three-moduli RSNS with any form of moduli as well as N -moduli RSNS.

3.5.1 Implementation Issues in RSNS

Selection of Moduli Set

In the design of RSNS encoder, the first task is to determine the PRP moduli set $\mathbf{m} = [m-1 \ m \ m+1]$ based on the system requirement. The smallest PRP moduli set of this form possible is $\mathbf{m} = [3 \ 4 \ 5]$. If the number system is required to represent Z unique integers, the moduli set must be selected such that dynamic range $\hat{M}$ is higher or equal to Z , i.e.

$$\begin{aligned} \hat{M} &\geq Z \\ \frac{3}{2}m^2 + \frac{9}{2}m + 1 &\geq Z \end{aligned} \tag{3.21}$$

$$m \geq \frac{\sqrt{57 + 24Z} - 9}{6} \tag{3.22}$$

where $m \geq 4$ and m even. This constraint gives the lower bound of m . Any value of m can be chosen as long as it satisfies the dynamic range requirement. However, as receiver complexity increases with the increase in dynamic range $\hat{M}$, the value of m must be kept as low as possible. Hence in practice, m is selected such that it is the smallest even number that satisfies equation (3.22).

Shift of Longest Sequence $\hat{M}$

In Section 3.2, it has been discussed that the position of longest residue sequence $\hat{M}$ in RSNS does not start from 0, but from the *start* to *end* position as calculated in equation (3.5). However, in most practical applications (such as digital transmission or communication system), it is often required to represent Z unique integers within

the range of $[0, Z)$ to minimize number of bits used for transmission. Hence, in this case, these integers need to be shifted to fall within the encoder's dynamic range $[0, Z)$ by using appropriate offset, i.e. the *start* position. The same holds true for the corresponding decoder. The result of the CRT computation also needs to be shifted back to obtain the transmitted integer. This sequence shift is illustrated in Figure 3.3.

Figure 3.3: Sequence shift in RSNS

Example: Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. As calculated before in Example 3.1 on page 42, the longest sequence is located at position *start* = 79 to *end* = 121, and the length is $\hat{M} = 43$. Assume now that we need to represent 32 unique integers from 0 to 31 (i.e. $0 \leq h \leq 31$). These integers need to be shifted such that they fall within the encoder's dynamic range $79 \leq X \leq 110$ before encoding. For example, if the transmitted integer is $h = 12$, this integer first needs to be shifted such that $X = 12 + 79 = 91$ and encoded by equation (3.4) into its corresponding RSNS residues $\mathbf{r} = [0 \ 2 \ 1]$. At the decoder, assuming correct reception, i.e. $\mathbf{r}' = \mathbf{r}$, the CRT computation of residue vector $\mathbf{r}' = [0 \ 2 \ 1]$ yields $X' = 91$. This result then needs to be shifted back to obtain the transmitted integer $h' = 91 - 79 = 12$.

Use of Look-up Table (LUT) for Decoding

The high complexity of conventional decoding due to the use of multiple CRTs, together with the short dynamic range of RSNS, suggests that the *decoder* can be

efficiently implemented using a *look-up table* (LUT). Furthermore, LUT can also be used at the *encoder* to minimize encoding time. In RSNS, the use of LUT is justifiable due to its short dynamic range.

Given that the RSNS needs to represent all integers h within the range of $0 \leq h < Z$ (Z is the number of unique integers required), the encoder and decoder compute all residue vectors $\mathbf{r}$ corresponding to the shifted integers X (where $X = h + \text{offset}$ and $\text{offset} \leq X < \text{offset} + Z$) and stores it in the LUT. Consequently, the LUT index corresponds to the desired integer h . This implies that *no further sequence shift* is required in the process of encoding and decoding for the given LUT.

3.5.2 RSNS Encoding and Decoding Mechanism using LUT

At the encoder, the transmitted integer h is used as an index search for the LUT. In the context of three-moduli RSNS, the LUT entry $\mathbf{r} = [r_1 \ r_2 \ r_3]$ corresponding to index h is transmitted.

At the decoder, the noise-perturbed received vector $\mathbf{r}' = [r'_1 \ r'_2 \ r'_3]$ is compared with each LUT entry, and the LUT entry corresponding to the received residue vector $\mathbf{r}'$ is taken to be the transmitted residue vector. Consequently, its position index h' is determined as the transmitted integer. This encoding and decoding mechanism is illustrated in Figure 3.4.

Figure 3.4: Encoding and decoding mechanism in RSNS

Example 3.3

Selection of Moduli Set. Assume all integers h within the range $0 \leq h \leq 31$ need to be represented in a three-moduli RSNS with moduli set of the form $[m-1 \ m \ m+1]$.

From equation (3.22), the moduli set must be selected such that

$$m \geq \frac{\sqrt{57 + 24 \times 32} - 9}{6} = 3.29$$

Thus, smallest moduli set that can be selected is $\mathbf{m} = [3 \ 4 \ 5]$.

Structure of LUT. As calculated previously in Example 3.1 on page 42, the longest sequence in an RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$ starts from $X = 79$ and ends at $X = 121$, and dynamic range $\hat{M} = 43$. In this case, since we need to represent 32 integers ($0 \leq h \leq 31$), residue vectors $\mathbf{r}$ corresponding to integers $79 \leq X \leq 110$ are stored in the LUT. Consequently, the LUT index shows integer h as shown in Table 3.10.

$r_1(m_1 = 3)$	2	2	3	3	3	2	2	2	1	1	1
$r_2(m_2 = 4)$	2	3	3	3	4	4	4	3	3	3	2
$r_3(m_3 = 5)$	3	3	3	2	2	2	1	1	1	0	0
X	79	80	81	82	83	84	85	86	87	88	89
h	0	1	2	3	4	5	6	7	8	9	10

$r_1(m_1 = 3)$	0	0	0	1	1	1	2	2	2	3	3
$r_2(m_2 = 4)$	2	2	1	1	1	0	0	0	1	1	1
$r_3(m_3 = 5)$	0	1	1	1	2	2	2	3	3	3	4
X	90	91	92	93	94	95	96	97	98	99	100
h	11	12	13	14	15	16	17	18	19	20	21

$r_1(m_1 = 3)$	3	2	2	2	1	1	1	0	0	0
$r_2(m_2 = 4)$	2	2	2	3	3	3	4	4	4	3
$r_3(m_3 = 5)$	4	4	5	5	5	4	4	4	3	3
X	101	102	103	104	105	106	107	108	109	110
h	22	23	24	25	26	27	28	29	30	31

Table 3.10: Encoding of RSNS using Look-up Table (LUT)

Encoding and Decoding. Now, assume that we want to transmit integer $h = 10$. Referring to the LUT in Table 3.10, index $h = 10$ is mapped to residue vector

$\mathbf{r} = [1 \ 2 \ 0]$ and transmitted. The received vector is the transmitted vector plus noise (error), i.e. $\mathbf{r}' = \mathbf{r} + \mathbf{e}$.

Consider the following possible received vectors:

- **Case 1.** Assuming perfect reception ($\mathbf{e} = \mathbf{0}$), the received vector is the same as transmitted vector, i.e. $\mathbf{r}' = \mathbf{r}$. The LUT index corresponding to the received vector $\mathbf{r}' = [1 \ 2 \ 0]$ is $h' = 10$. In this case, integer h' is correctly decoded.
- **Case 2.** Now, consider the situation where the transmitted vector is perturbed by noise with error pattern $\mathbf{e} = [-1 \ 0 \ 0]$. The received vector is now $\mathbf{r}' = [0 \ 2 \ 0]$. In this case, the decoder finds a matching entry in the LUT in Table 3.10; the received vector $\mathbf{r}'$ corresponds to integer $h' = 11$. Although the received vector is incorrect, the error is *undetected* as the received vector is *valid*.
- **Case 3.** Next, consider another situation where the transmitted vector is perturbed by noise where $\mathbf{e} = [0 \ 0 \ 1]$. The received vector is now $\mathbf{r}' = [1 \ 2 \ 1]$. In this case, the decoder cannot find a matching entry in the LUT in Table 3.10. An *invalid* residue vector is received and error is *detected*.

The above example shows that the received residue vector can either be *valid* if it exists in the LUT table (Case 1,2) or *invalid* otherwise (Case 3). If the received residue vector $\mathbf{r}'$ exists in the LUT, its position index h' is determined as the transmitted integer. If the $\mathbf{r}'$ does not exist in the LUT, an error is *detected*.

This implies that RSNS by itself can be used for error detection *without* adding any redundancy as in RNS. RSNS residue sequence by itself is inherently redundant, hence it makes the dynamic range much shorter compared to RNS. Besides, RSNS is also potential to be used for error correction, provided that efficient error

correction algorithm can be devised.

3.6 Summary

In this chapter, the encoding, decoding as well as the dynamic range of RSNS have been discussed. Due to the decoder complexity and the short dynamic range of RSNS, the use of a lookup table (LUT) is recommended. RSNS also inherently shows some potential for error detection and correction without any redundancy. In the following chapter, the application of RSNS in error control is explored.

Chapter 4

Application of RSNS for Error Control

This chapter proposes the application of RSNS for error detection and correction. As opposed to RNS, RSNS by itself can be used for error detection *without* adding any redundancy. This attractive feature is explored further for use in error control.

4.1 Error Detection in RSNS

In RSNS, error detection is achieved by the non-existence of possible received residue vectors in the LUT. In other words, error is detected if and only if the received residue vector $\mathbf{r}'$ does not exist in the LUT. In the following discussion, these permutations of received vectors that do not exist in the LUT are referred to as *holes*.

4.1.1 Definition of Holes in RSNS

A *hole* can be defined as the non-existence of valid residue vectors within the possible RSNS dynamic range $\hat{M}$. This non-existence is caused by noise perturbation in

the transmission channel, causing a valid vector to fall outside the dynamic range. If a hole is received at the receiver, transmission error has occurred, and error is successfully *detected*. However, if a valid residue vector is received, nothing can be concluded — it can either be an errorless transmission or an undetectable error caused by valid vector that does not fall into a hole after noise perturbation.

Therefore, there are two types of transmission errors in RSNS:

- Errors that are successfully *detected* if the received vector falls into holes (i.e. $\mathbf{r}' \neq \mathbf{r} : r \in \text{LUT}, r' \notin \text{LUT}$), or
- Errors that are *undetected* if the received vector exists in the LUT and different from the transmitted vector (i.e. $\mathbf{r}' \neq \mathbf{r} : r, r' \in \text{LUT}$)

In other words,

$$P_E = P_D + P_U \quad (4.1)$$

where P_E is the probability of error, P_D is the probability of detected error and P_U is the probability of undetected error.

4.1.2 Analysis of Error Detection Probability in RSNS

In RSNS, error detection probability increases with the increase in the number of holes present. Consider a three-moduli RSNS with moduli set $\mathbf{m} = [m_1 \ m_2 \ m_3]$. Since each residue r_i can take up any values from 0 to m_i , if a uniform k -bit binary transmission is to be used for each residue channel, k needs to satisfy the following constraint:

$$k \geq \lceil \log_2(\max(m_i + 1)) \rceil \quad (4.2)$$

where $i = 1, 2, 3$. If moduli set is chosen to be of the form $\mathbf{m} = [m-1 \ m \ m+1]$, then $k \geq \lceil \log_2(m+2) \rceil$.

Since LUT is used in the proposed system, after noise perturbation, each received residue r'_i can take any value between 0 and $2^k - 1$, i.e. $r'_i \in \{0, \dots, 2^k - 1\}$. Thus, the total number of representable integers in each residue channel at the receiver is 2^k , and the total representable range of the received residue *vector* (sample space) is $\mathcal{S} = 2^{3k}$. Hence, the number of holes Δ is the difference between the representable range of the received residue vector and the dynamic range of RSNS sequence, i.e.

$$\Delta = \mathcal{S} - \hat{M} \quad (4.3)$$

$$= 2^{3k} - \left(\frac{3}{2}m^2 + \frac{9}{2}m + 1 \right) \quad (4.4)$$

From equation (4.1), the upper bound of error detection probability ($P_D = 1$) is reached when all transmission errors are successfully *trapped* in the holes, i.e. the probability of undetected error is zero ($P_U = 0$). The expected error detection probability is thus equivalent to the probability of holes in the sample space of the received residue vector, and is given by

$$\mathbb{E}[P_D] = \frac{\Delta}{\mathcal{S}} = 1 - \frac{\frac{3}{2}m^2 + \frac{9}{2}m + 1}{2^{3k}} \quad (4.5)$$

The number of holes Δ and the expected error detection probability $\mathbb{E}[P_D]$ of several PRP moduli sets of the form $[m-1 \ m \ m+1]$ is summarized in Table 4.1.

From Table 4.1, it can be seen that RSNS has a very good inherent error detection capability due to its short dynamic range ($\mathbb{E}[P_D] > 0.9$) for any moduli set, i.e. most errors can be successfully detected). Also, from equation (4.4), as number of bits per residue k is increased, the representable range $\mathcal{S}$ and the number of holes

$\mathbf{m}$	$\hat{M}$	Sample Space $\mathcal{S}$			Number of Holes Δ			$E[P_D](\text{in}\%)$		
		$k = 4$	$k = 5$	$k = 6$	$k = 4$	$k = 5$	$k = 6$	$k = 4$	$k = 5$	$k = 6$
[61 62 63]	6046	—	—	262144	—	—	256098	—	—	97.69
[51 52 53]	4291	—	—	262144	—	—	257853	—	—	98.36
[41 42 43]	2836	—	—	262144	—	—	259308	—	—	98.92
[29 30 31]	1486	—	32768	262144	—	31282	260658	—	95.47	99.43
[23 24 25]	973	—	32768	262144	—	31795	261171	—	97.03	99.63
[17 18 19]	568	—	32768	262144	—	32200	261576	—	98.27	99.78
[13 14 15]	358	4096	32768	262144	3738	32410	261786	91.26	98.90	99.86
[9 10 11]	196	4096	32768	262144	3900	32572	261948	95.21	99.40	99.93
[7 8 9]	133	4096	32768	262144	3963	32635	262011	96.75	99.59	99.95
[3 4 5]	43	4096	32768	262144	4053	32725	262101	98.95	99.87	99.98

Table 4.1: Number of holes in RSNS

Δ increase exponentially, thus resulting in an increasing expected probability of detection $E[P_D]$.

This theoretical analysis on the *expected* probability of detection is valid under the assumption that after noise perturbation, all possible received residue vectors $\mathbf{r}'$ in the sample space $\mathcal{S}$ are equiprobable. Unfortunately, due to the *integer Gray coding* property of RSNS, this assumption may not be true in practice. Since single bit errors are more likely to occur than burst errors in a transmission channel [4], perturbation in *one* residue channel causes a residue vector $\mathbf{r}$ to fall into $\mathbf{r}'$ that has a high similarity or correlation with $\mathbf{r}$. Besides, a high correlation exists between RSNS residue vectors of two *consecutive* integers, since they only differ by ± 1 in one of their residue values. This results in a higher probability of undetected error P_U (as in Case 2 of Example 3.3 on page 52), and by equation (4.1), the actual error detection probability P_D is lower.

The following section studies the integer Gray property of RSNS sequence.

4.1.3 Integer Gray Code property of RSNS

One distinct feature that RSNS possesses over RNS is its *Gray-code* property, in which only one MRS value changes from one integer h to the neighboring symbol $h+1$, and this MRS value varies by ± 1 . Consider a three-moduli RSNS with moduli

set $\mathbf{m} = [3 \ 4 \ 5]$. All residue vectors within dynamic range $\hat{M}$ are shown in Table 4.2 (derived in Example 3.3 on page 52).

$r_1(m_1 = 3)$	2	2	3	3	3	2	2	2	1	1	1	0	0	0	1
$r_2(m_2 = 4)$	2	3	3	3	4	4	4	3	3	3	2	2	2	1	1
$r_3(m_3 = 5)$	3	3	3	2	2	2	1	1	1	0	0	0	1	1	1
h	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$r_1(m_1 = 3)$	1	1	2	2	2	3	3	3	2	2	2	1	1	1	
$r_2(m_2 = 4)$	1	1	0	0	1	1	1	2	2	2	3	3	3	4	
$r_3(m_3 = 5)$	2	2	2	3	3	3	4	4	4	5	5	5	4	4	
h	15	16	17	18	19	20	21	22	23	24	25	26	27	28	

$r_1(m_1 = 3)$	0	0	0	1	1	1	2	2	2	3	3	3	2	2	
$r_2(m_2 = 4)$	4	4	3	3	3	2	2	2	1	1	1	0	0	0	
$r_3(m_3 = 5)$	4	3	3	3	2	2	2	1	1	1	0	0	0	1	
h	29	30	31	32	33	34	35	36	37	38	39	40	41	42	

Table 4.2: Three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$

From Table 4.2, it can be seen that the residue vector of integer $h = 0$ differs with the next integer $h = 1$ in the *second* MRS value. Subsequently, the residue vector of integer $h = 1$ differs with $h = 2$ in the *first* MRS value, and the residue vector of integer $h = 2$ differs with $h = 3$ in the *third* MRS value, and so on. This sequence of change in the position of MRS values repeats again in the next cycle. In this case, an MRS value only changes after a period of $N = 3$, where N is the number of modulus in the moduli set.

Conversely speaking, if an integer h changes to integer $h + 1$ in the i^{th} MRS value, integer $h + 1$ will *not* change to $h + 2$ in the i^{th} MRS again. In other words, if an integer h corresponds to residue vector $\mathbf{r}_h = [r_1 \ r_2 \ r_3]$, and assuming h changes to the next integer $h + 1$ in the *second* MRS value, integer $h + 1$ would correspond to vector $\mathbf{r}_{h+1} = [r_1 \ r_2 \pm 1 \ r_3]$. And since the transition from integer $h + 1$ to $h + 2$ does not occur in the same MRS again, $h + 2$ will *not* correspond to residue vector $\mathbf{r}_{h+2} = [r_1 \ r_2 \pm 2 \ r_3]$. Also, due to the PRP property of the moduli values and considering that dynamic range of RSNS $\hat{M}$ is relatively short, it is highly unlikely that residue vector $[r_1 \ r_2 \pm 2 \ r_3]$ exists within $\hat{M}$.

For example, consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$ in Table 4.2. Integer $h = 0$ corresponds to $\mathbf{r} = [2 \ 2 \ 3]$ and integer $h = 1$ corresponds to $\mathbf{r} = [2 \ 3 \ 3]$. This transition involves a change in the second MRS value. It can also be seen that residue vector $\mathbf{r} = [2 \ 4 \ 3]$ does not exist in the LUT, and is therefore a *hole*.

In other words, there is a *high* correlation between a residue vector $\mathbf{r} = [r_1 \ r_2 \ r_3]$ and another vector that differs in a single MRS value than itself (i.e. $[r_1 \pm 1 \ r_2 \ r_3]$, $[r_1 \ r_2 \pm 1 \ r_3]$ and $[r_1 \ r_2 \ r_3 \pm 1]$), but a much lower correlation with other vectors that are far apart from itself.

This unequal correlation among residue vectors results in a higher probability of undetected error P_U if the residue vectors that have high correlation values frequently exist in the LUT. Conversely speaking, it is *possible* to devise an *optimal* system which is able to isolate the residue vectors that have high correlation values and only maintain those with low correlation values in the LUT. Hence, it is *not impossible* to design a system that can have a *better* error detection probability than the expected value $E[P_D]$, or even a perfect error detection probability of $P_D = 1$.

This analysis suggests that *mapping* is crucial in RSNS. Efficient binary representation schemes can help to make the performance of RSNS better. Different mapping schemes have different correlation values due to different Hamming distances between two consecutive integers, and result in a different error detection performance. In the following sections, three binary representation schemes are introduced for RSNS. They are based on binary code, Gray code and inverse Gray code representations. Among these three schemes, binary and Gray codes are widely known and used in many applications today. Inverse Gray code is a *novel* binary mapping technique and will be elaborated in the later sections.

4.2 Binary Representation for RSNS: (1) Binary Code

4.2.1 Encoding and Decoding of Binary Code

Nowadays, binary code is used by virtually all modern computer systems. An integer h is represented with a binary code $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$, where

$$b_j = \left\lfloor \frac{h - \sum_{i=1}^{j-1} b_i \cdot 2^{k-i}}{2^{k-j}} \right\rfloor \quad (4.6)$$

and $j = 1, 2, \dots, k$. Example of four-bit binary code ($k = 4$) is shown in Table 4.3.

Integer	Binary Code $\mathbf{b}$	Gray Code $\mathbf{g}$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Table 4.3: Four-bit binary and Gray codes

Integer h that corresponds to a k -bit binary code $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$ can be decoded by

$$h = \sum_{i=1}^k b_i \cdot 2^{k-i} \quad (4.7)$$

4.2.2 Distance Properties of Binary-coded Integers

It can be observed from Table 4.3 that binary code for an integer h differs by one bit from the next integer $h+1$ if h is even, and by at least two bits if h is odd, i.e. $d(\mathbf{b}_h, \mathbf{b}_{h\pm 1}) \geq 1$.

Analyzing the four-bit binary code in Table 4.3, the Hamming distance between two binary codes corresponding to integers h_1 and h_2 ($0 \leq h_1, h_2 \leq 15$), i.e. $d(\mathbf{b}_{h_1}, \mathbf{b}_{h_2})$ can be found and summarized in Table 4.4.

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	4
1	1	0	2	1	2	1	3	2	2	1	3	2	3	2	4	3
2	1	2	0	1	2	3	1	2	2	3	1	2	3	4	2	3
3	2	1	1	0	3	2	2	1	3	2	2	1	4	3	3	2
4	1	2	2	3	0	1	1	2	2	3	3	4	1	2	2	3
5	2	1	3	2	1	0	2	1	3	2	4	3	2	1	3	2
6	2	3	1	2	1	2	0	1	3	4	2	3	2	3	1	2
7	3	2	2	1	2	1	1	0	4	3	3	2	3	2	2	1
8	1	2	2	3	2	3	3	4	0	1	1	2	1	2	2	3
9	2	1	3	2	3	2	4	3	1	0	2	1	2	1	3	2
10	2	3	1	2	3	4	2	3	1	2	0	1	2	3	1	2
11	3	2	2	1	4	3	3	2	2	1	1	0	3	2	2	1
12	2	3	3	4	1	2	2	3	1	2	2	3	0	1	1	2
13	3	2	4	3	2	1	3	2	2	1	3	2	1	0	2	1
14	3	4	2	3	2	3	1	2	2	3	1	2	1	2	0	1
15	4	3	3	2	3	2	2	1	3	2	2	1	2	1	1	0

Table 4.4: Hamming distance d of four-bit binary code

In the following discussion, *neighbors* of an integer are defined as the integers whose representation differs by one bit from itself. Thus in general, a k -bit code has k neighbors. For example, in a four-bit binary representation, integer $h = 0 = [0\ 0\ 0\ 0]_2$ has four neighbors, i.e. $1 = [0\ 0\ 0\ 1]_2$, $2 = [0\ 0\ 1\ 0]_2$, $4 = [0\ 1\ 0\ 0]_2$ and $8 = [1\ 0\ 0\ 0]_2$. It can be verified from the first row of the distance matrix in Table 4.4 that these four integers have Hamming distance $d = 1$ with integer $h = 0$.

Table 4.5 shows the list of neighbors of each integer h in a four-bit binary system, where $0 \leq h \leq 15$. The bold-faced integers show the neighbors that differ by ± 1 from integer h .

Integer h	Neighbors			
0	1	2	4	8
1	0	3	5	9
2	0	3	6	10
3	1	2	7	11
4	0	5	6	12
5	1	4	7	13
6	2	4	7	14
7	3	5	6	15
8	0	9	10	12
9	1	8	11	13
10	2	8	11	14
11	3	9	10	15
12	4	8	13	14
13	5	9	12	15
14	6	10	12	15
15	7	11	13	14

Table 4.5: Neighbors of four-bit binary code

4.2.3 Error Detection using Binary RSNS

In an N -moduli RSNS, there are four types of errors in terms of the number of bits and MRSs affected:

- Single bit error in single MRS
- Single bit error in multiple MRSs
- Multiple bit error (burst error) in single MRS
- Multiple bit error (burst error) in multiple MRSs

In this report, single bit error in single and multiple MRSs will be analyzed. Due to the analytical complexity of all possible permutations of burst error, the third and fourth case are excluded from the scope of this report. A justification for this assumption is that the probability of occurrence of i errors is less than the probability of occurrence of j errors if $i > j$ [4]. Since an integer h is split into N residues in an N -moduli RSNS and transmitted independently in N channels, the occurrence

of single bit error affecting one or more MRSs is more likely compared to burst error. In this context, since we are using three-moduli RSNS, the following analysis of error detection is divided into three cases — single bit error in 1, 2 and 3 MRSs.

Case 1: Single bit error in one MRS

In this type of error, single bit error occurs in one of the MRS values (either in the first, second or third MRS), and causes the integer residue in that MRS to fall into its one of its neighbors. An example is provided as follows.

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *one* MRS, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.6.

$\mathbf{r}$	2											
	2											
	3											
h	0											

 $\rightarrow$

$\mathbf{r}'$	0	3	6	10	2	2	2	2	2	2	2	2
	2	2	2	2	0	3	6	10	2	2	2	2
	3	3	3	3	3	3	3	3	1	2	7	11
h'	$\times$	$\times$	$\times$	$\times$	18	1	$\times$	$\times$	36	35	$\times$	$\times$

Table 4.6: Binary RSNS — Single bit error in *one* MRS for $\mathbf{r} = [2 \ 2 \ 3]$

Among 12 possible received residue vectors in Table 4.6, it can be seen that error cannot be detected in four residue vectors, i.e. $\mathbf{r}' = [2 \ 0 \ 3]$, $[2 \ 3 \ 3]$, $[2 \ 2 \ 1]$ and $[2 \ 2 \ 2]$. These four vectors do not fall into holes and correspond to integers $h' = 18, 1, 36$ and 35 respectively. Thus in this case, $P_U = \frac{4}{12} = 0.33$ and $P_D = \frac{8}{12} = 0.67$. This P_D is still much lower compared to the average probability of detection given in Table 4.1, i.e. $\overline{P_D} = 0.9895$. The aim of the proposed research is to make P_D as high as possible by trapping all received residue vectors caused by single bit errors in one or more MRSs into holes so that all errors can be successfully detected.

Case 2: Single bit error in two MRSs

In this type of error, single bit error occurs in two MRSs. In a three-moduli RSNS, these errors occur either in the 1st and 2nd, 1st and 3rd, or 2nd and 3rd MRSs.

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *two* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.7.

$\mathbf{r}$	2	$\rightarrow$															
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
h	0	42	17	$\times$	$\times$	7	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	2	7	11	1	2	7	11	1	2	7	11	1	2	7	11	11
h'	12	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
2	0	3	6	10	0	3	6	10	0	3	6	10	0	3	6	10	10
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
h'	$\times$	31	$\times$	$\times$	$\times$	2	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.7: Binary RSNS — Single bit error in *two* MRSs for $\mathbf{r} = [2 \ 2 \ 3]$

From Table 4.7, it is seen that among 48 possible received residue vectors, error cannot be detected in six residue vectors. Hence, $P_U = \frac{6}{48} = 0.125$ and $P_D = 1 - P_U = 0.875$. Similar with Case 1 (single bit error in one MRS), this P_D is still lower compared to the average probability of detection given in Table 4.1, i.e. $\overline{P_D} = 0.9895$. However, it is also noted that P_D in Case 2 is higher than that of Case 1. This is expected, because as the number of erroneous MRSs increases, the likelihood of falling into holes is higher due to the lower correlation between two residue vectors that are located far apart.

Case 3: Single bit error in three MRSs

In this type of error, single bit error occurs in three MRSs. In a three-moduli RSNS, this refers to single bit error in all MRSs.

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *all* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.8.

$\mathbf{r}$	2																
	2																
	3																
h	0																

→

$\mathbf{r}'$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
	1	2	7	11	1	2	7	11	1	2	7	11	1	2	7	11	11
h'	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

$\mathbf{r}'$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
	1	2	7	11	1	2	7	11	1	2	7	11	1	2	7	11	11
h'	×	×	×	×	×	3	×	×	×	×	×	×	×	×	×	×	×

$\mathbf{r}'$	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
	1	2	7	11	1	2	7	11	1	2	7	11	1	2	7	11	11
h'	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

$\mathbf{r}'$	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	0	0	0	0	3	3	3	3	6	6	6	6	10	10	10	10	10
	1	2	7	11	1	2	7	11	1	2	7	11	1	2	7	11	11
h'	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

Table 4.8: Binary RSNS — Single bit error in *all* MRSs for $\mathbf{r} = [2 \ 2 \ 3]$

It can be seen from Table 4.8 that among 64 possible received residue vectors, error cannot be detected in one residue vectors, i.e. $\mathbf{r}' = [3 \ 3 \ 2]$ corresponding to integer $h' = 3$. Hence, $P_U = \frac{1}{64} = 0.015625$ and $P_D = 1 - P_U = 0.984375$. Again, the probability of detection in this case is higher compared to Case 1 and 2. This shows that binary RSNS performs better as the number of erroneous MRSs increases.

4.3 Binary Representation for RSNS: (2) Gray Code

The *Gray code* (also called *reflected binary code*), named after its inventor Frank Gray, is a binary numeral system where two successive integers differ in only one bit (i.e. Hamming distance of two consecutive integers h and $h \pm 1$ is one). Gray code is cyclic in nature, that is, the first and last values of the sequence differ by

only one bit.

Gray code is widely used in angle-measuring devices, genetic algorithms, Karnaugh maps, and most importantly, in digital communications. In digital communications, Gray code plays an important role in error correction. For example, in a digital modulation scheme such as QAM where data is typically transmitted in symbols of 4 bits or more, the signal's constellation diagram is arranged so that the bit patterns conveyed by adjacent constellation points differ by only one bit. By combining this with forward error correction capable of correcting single bit errors, it is possible for a receiver to correct any transmission errors that cause a constellation point to deviate into the area of an adjacent point. This makes the transmission system less susceptible to noise.

These attractive features of Gray code make us look more into its potential to be implemented in the context of RSNS to give better error detection and correction.

4.3.1 Encoding and Decoding of Gray Code

A k -bit Gray code $\mathbf{g} = [g_1 \ g_2 \ \dots \ g_k]$ can be *encoded* from integer h with binary representation $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$ as follows:

- Initialize $g_1 = b_1$.
- For $1 < i \leq k$, $g_i = b_{i-1} \oplus b_i$, where $\oplus$ denotes logical XOR operation.

Following this encoding rule, four-bit Gray code ($k = 4$) can be obtained and shown in Table 4.3 on page 61.

Integer h that corresponds to a k -bit binary code $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$ can be *decoded* from Gray code $\mathbf{g} = [g_1 \ g_2 \ \dots \ g_k]$ as follows:

- Initialize $b_1 = g_1$.
- For $1 < i \leq k$, $b_i = b_{i-1} \oplus g_i$, where $\oplus$ denotes logical XOR operation.

4.3.2 Distance Properties of Gray-coded Integers

It can be observed from Table 4.3 that Hamming distance of Gray code for two consecutive integers h and $h \pm 1$ is one, i.e. $d(\mathbf{g}_h, \mathbf{g}_{h \pm 1}) = 1$.

Analyzing the four-bit Gray code in Table 4.3, the Hamming distance between two Gray codes corresponding to integers h_1 and h_2 ($0 \leq h_1, h_2 \leq 15$), i.e. $d(\mathbf{b}_{h_1}, \mathbf{b}_{h_2})$ can be found and summarized in Table 4.9.

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	1	2	3	2	1	2	3	4	3	2	3	2	1
1	1	0	1	2	3	2	1	2	3	2	3	4	3	2	1	2
2	2	1	0	1	2	1	2	3	4	3	2	3	2	1	2	3
3	1	2	1	0	1	2	3	2	3	4	3	2	1	2	3	2
4	2	3	2	1	0	1	2	1	2	3	2	1	2	3	4	3
5	3	2	1	2	1	0	1	2	3	2	1	2	3	2	3	4
6	2	1	2	3	2	1	0	1	2	1	2	3	4	3	2	3
7	1	2	3	2	1	2	1	0	1	2	3	2	3	4	3	2
8	2	3	4	3	2	3	2	1	0	1	2	1	2	3	2	1
9	3	2	3	4	3	2	1	2	1	0	1	2	3	2	1	2
10	4	3	2	3	2	1	2	3	2	1	0	1	2	1	2	3
11	3	4	3	2	1	2	3	2	1	2	1	0	1	2	3	2
12	2	3	2	1	2	3	4	3	2	3	2	1	0	1	2	1
13	3	2	1	2	3	2	3	4	3	2	1	2	1	0	1	2
14	2	1	2	3	4	3	2	3	2	1	2	3	2	1	0	1
15	1	2	3	2	3	4	3	2	1	2	3	2	1	2	1	0

Table 4.9: Hamming distance d of four-bit Gray code

Table 4.10 shows the list of neighbors of each integer h in a four-bit Gray system, where $0 \leq h \leq 15$. The bold-faced integers show the neighbors that differ by ± 1 from integer h .

4.3.3 Error Detection using Gray RSNS

Similar with the previous error analysis of binary RSNS, error detection in Gray RSNS can also be divided into three cases — single bit error in 1, 2 and 3 MRSs.

Integer h	Neighbors			
0	1	3	7	15
1	0	2	6	14
2	1	3	5	13
3	0	2	4	12
4	3	5	7	11
5	2	4	6	10
6	1	5	7	9
7	0	4	6	8
8	7	9	11	15
9	6	8	10	14
10	5	9	11	13
11	4	8	10	12
12	3	11	13	15
13	2	10	12	14
14	1	9	13	15
15	0	8	12	14

Table 4.10: Neighbors of four-bit Gray code

Case 1: Single bit error in one MRS

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *one* MRS, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.11.

$\mathbf{r}$	2												
	2												
	3												
h	0												

 $\rightarrow$

$\mathbf{r}'$	1	3	5	13	2	2	2	2	2	2	2	2	2
	2	2	2	2	1	3	5	13	2	2	2	2	2
	3	3	3	3	3	3	3	3	0	2	4	12	
h'	$\times$	$\times$	$\times$	$\times$	19	1	$\times$	$\times$	$\times$	35	23	$\times$	

Table 4.11: Gray RSNS — Single bit error in *one* MRS for $\mathbf{r} = [2 \ 2 \ 3]$

Among 12 possible received residue vectors in Table 4.11, it can be seen that error cannot be detected in four residue vectors, i.e. $\mathbf{r}' = [2 \ 1 \ 3], [2 \ 3 \ 3], [2 \ 2 \ 2]$ and $[2 \ 2 \ 4]$ corresponding to integers $h' = 19, 1, 35$ and 23 respectively. Thus, $P_U = \frac{4}{12} = 0.33$ and $P_D = \frac{8}{12} = 0.67$.

Case 2: Single bit error in two MRSs

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *two* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.12.

$\mathbf{r}$	2																
	2																
	3																
h	0																

→

$\mathbf{r}'$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

$\mathbf{r}'$	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	10	34	×	×	×	×	×	22	×	×	×	×	×	×	×	×	×

$\mathbf{r}'$	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	1	3	5	13	1	3	5	13	1	3	5	13	1	3	5	13	13
	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
h'	×	32	×	×	20	2	×	×	×	×	×	×	×	×	×	×	×

Table 4.12: Gray RSNS — Single bit error in *two* MRSs for $\mathbf{r} = [2 \ 2 \ 3]$

From Table 4.12, it can be seen that among 48 possible received residue vectors, error cannot be detected in six residue vectors. Hence, $P_U = \frac{6}{48} = 0.125$ and $P_D = 1 - P_U = 0.875$.

Case 3: Single bit error in three MRSs

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *all* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.13.

From Table 4.13, it can be seen that among 64 possible received residue vectors, error cannot be detected in seven residue vectors. Hence, $P_U = \frac{7}{64} = 0.109375$ and $P_D = 1 - P_U = 0.890625$.

$\mathbf{r}$	2																
	2																
	3																
h	0																

 $\rightarrow$

$\mathbf{r}'$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	$\times$	15	$\times$	$\times$	9	33	27	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	39	$\times$	21	$\times$	$\times$	$\times$	3	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
	1	1	1	1	3	3	3	3	5	5	5	5	13	13	13	13	13
	0	2	4	12	0	2	4	12	0	2	4	12	0	2	4	12	12
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.13: Gray RSNS — Single bit error in *all* MRSs for $\mathbf{r} = [2\ 2\ 3]$

4.3.4 Suitability of Binary and Gray Code for RSNS

In general, RSNS incorporating binary and Gray codes do not perform very well in terms of error detection due to the high probability of undetected error. From the list of neighbors of binary and Gray-coded integers in Table 4.5 and 4.10, it can be observed that every integer h always has another integer(s) which differ by ± 1 as its neighbor(s), thus causing a single bit perturbation in an integer h to fall into $h \pm 1$.

In the context of RSNS, this property directly translates to undetected errors due to the integer Gray property of RSNS sequence. A single bit perturbation in one MRS causes a residue vector $\mathbf{r}$ corresponding to integer h to fall into another residue vector $\mathbf{r}'$ that corresponds to integer $h' = h \pm 1$. This can be verified from single bit error perturbations in Table 4.6 and 4.11 that integer $h' = h \pm 1$ *always* appears as one of the residue vectors that causes undetected errors. To improve the error detection probability of RSNS, this problem needs to be solved as it gives some sort of *lower bound* to the probability of undetected error.

In fact, from the above examples, it can be seen that Gray code generally performs poorer compared to binary code in the context of RSNS. This is because in Gray code, the Hamming distance between two consecutive integers is always one,

while for binary code, the Hamming distance between integer h and $h+1$ is one if h is even and at least two if h is odd. Thus, the closer the Hamming distance, the poorer the performance. In the following section, a novel binary representation called *Inverse Gray* is proposed to improve the error detection capability of RSNS. This method basically tries to maximize the Hamming distance between two consecutive integers so that undetectable errors can be minimized.

A conceptually similar mapping method called *Anti Gray Coding* (AGC) has been introduced in the literature such as in [38] and [39]. To the best of our knowledge, the existing literature on AGC mostly addresses issues related to image signal processing, vector quantization, and *limited* applications in communications. AGC is non-binary and non-deterministic in nature. Iterative computation needs to be carried out to obtain the anti-gray representation of an integer value [38], which may be relatively time-consuming in practice. On the other hand, the proposed inverse Gray mapping is simpler, non-iterative and exhibits better error detection and correction properties for systems such as RSNS based communication systems.

4.4 Binary Representation for RSNS: (3) Inverse Gray Code

As the name suggests, *Inverse Gray* is a binary representation whose mapping rule is the exact inverse (or opposite) of Gray code. In Gray code, two consecutive integers differ by one bit, i.e. Hamming distance is 1 for two consecutive integers. In Inverse Gray, two consecutive k -bit integers differ by $k-1$ bits; in other words, only one bit is *retained* in each transition between integer h and $h\pm 1$.

By using this single definition, there are many ways to construct an inverse Gray code by varying a bit to be retained in each transition between two consecutive

integers. For simplicity, in this research, we use the same positioning rule as Gray code. The position of the bit that *changes* in the Gray code is similar to the position of the bit that *does not change* in inverse Gray code.

Following this method, four-bit inverse Gray codes ($k = 4$) can be derived from Gray code as shown in Table 4.14. The bold-faced integers show the bit change between integer h and $h+1$.

Integer	Gray Code g	Inverse Gray Code v
0	0 0 0 0	0 0 0 0
1	0 0 0 1	1 1 1 0
2	0 0 1 1	0 0 1 1
3	0 0 1 0	1 1 0 1
4	0 1 1 0	0 1 1 0
5	0 1 1 1	1 0 0 0
6	0 1 0 1	0 1 0 1
7	0 1 0 0	1 0 1 1
8	1 1 0 0	1 1 0 0
9	1 1 0 1	0 0 1 0
10	1 1 1 1	1 1 1 1
11	1 1 1 0	0 0 0 1
12	1 0 1 0	1 0 1 0
13	1 0 1 1	0 1 0 0
14	1 0 0 1	1 0 0 1
15	1 0 0 0	0 1 1 1

Table 4.14: Inverse Gray code as derived from Gray code

For comparison, a graphical illustration of four-bit Gray and inverse Gray rotary encoder is given in Figure 4.1. The innermost circle represents the MSB, growing layer by layer until the outermost circle which represents the LSB. The white region denotes bit 0, while the shaded region denotes bit 1.

Similar encoding method can be extended for integer Gray codes with odd number of bits. However, it can be shown that unlike the case where number of bits k is even, the same mapping rule does not result in a unique code for integers within the representable range of $0 \leq h < 2^k$. Instead, it only gives unique codes for integers at the top half of the table, i.e. $0 \leq h < 2^{k-1}$. The inverse Gray code representing integers at the bottom half of the table ($2^{k-1} \leq h < 2^k$) is similar with the top half

Figure 4.1: Four-bit rotary encoder for Gray and inverse Gray code

of the table. A simple method of overcoming this non-uniqueness is by toggling the MSB of the inverse Gray code at the bottom half of the table.

An example for three-bit inverse Gray codes ($k = 3$) is shown in Table 4.15.

Integer	Gray code	Inverse Gray code		Inverse Gray code
0	0 0 0	0 0 0		0 0 0
1	0 0 1	1 1 0		1 1 0
2	0 1 1	0 1 1		0 1 1
3	0 1 0	1 0 1		1 0 1
4	1 1 0	1 1 0	→	0 1 0
5	1 1 1	0 0 0	→	1 0 0
6	1 0 1	1 0 1	→	0 0 1
7	1 0 0	0 1 1	→	1 1 1

Table 4.15: Three-bit inverse Gray code

Thus, the mapping rule for inverse Gray code is different for even and odd number of bits. The following section gives a complete encoding and decoding algorithm of inverse Gray code.

4.4.1 Encoding of Inverse Gray Code

A k -bit Inverse Gray code $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_k]$ can be derived from integer h with binary representation $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$ as follows:

- If number of bits k is even, initialize $b_0 = 0$. Else, initialize $b_0 = b_1$.

- If integer h is even ($b_k = 0$), $v_i = b_{i-1} \oplus b_i$, where $i = 1, \dots, k$ and $\oplus$ denotes logical XOR operation. If integer h is odd ($b_k = 1$), $v_i = \overline{b_{i-1} \oplus b_i}$, where $\oplus$ denotes logical XNOR operation.

This encoding algorithm is illustrated in the flowchart in Figure 4.2.

Example 1: In a four-bit binary representation, integer $h = 7$ is represented by $\mathbf{b} = [0 \ 1 \ 1 \ 1]$. Its Inverse Gray code can be obtained as follows:

- Since the number of bits k is even ($k = 4$), initialize $b_0 = 0$.
- Since the integer h is odd ($b_4 = 1$), using $v_i = \overline{b_{i-1} \oplus b_i}$, we have,

$$v_1 = \overline{b_0 \oplus b_1} = \overline{0 \oplus 0} = 1$$

$$v_2 = \overline{b_1 \oplus b_2} = \overline{0 \oplus 1} = 0$$

$$v_3 = \overline{b_2 \oplus b_3} = \overline{1 \oplus 1} = 1$$

$$v_4 = \overline{b_3 \oplus b_4} = \overline{1 \oplus 1} = 1$$

Thus, the four-bit inverse Gray representation of integer $h = 7$ is $\mathbf{v} = [1 \ 0 \ 1 \ 1]$.

Example 2: In a three-bit binary representation, integer $h = 6$ is represented by $\mathbf{b} = [1 \ 1 \ 0]$. Its Inverse Gray code can be obtained as follows:

- Since the number of bits k is odd, initialize $b_0 = b_1 = 1$.
- Since the integer h is even ($b_3 = 0$), using $v_i = b_{i-1} \oplus b_i$, we have,

$$v_1 = b_0 \oplus b_1 = 1 \oplus 1 = 0$$

$$v_2 = b_1 \oplus b_2 = 1 \oplus 1 = 0$$

$$v_3 = b_2 \oplus b_3 = 1 \oplus 0 = 1$$

Thus, the three-bit inverse Gray representation of integer $h = 6$ is $\mathbf{v} = [0 \ 0 \ 1]$.

4.4.2 Decoding of Inverse Gray Code

Integer h that corresponds to a k -bit binary code $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_k]$ can be decoded from inverse Gray code $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_k]$ as follows:

- If number of bits k is even, initialize $b_0 = 0$. Else, initialize $b_0 = v_1 \oplus v_2 \oplus \dots \oplus v_k$.
- If integer h is even (indicated by $v_1 \oplus \dots \oplus v_k = 0$ if k is even and $v_1 = 0$ if k is odd), $b_i = b_{i-1} \oplus v_i$. If integer h is odd (indicated by $v_1 \oplus \dots \oplus v_k = 1$ if k is even and $v_1 = 1$ if k is odd), $b_i = \overline{b_{i-1} \oplus v_i}$.

This decoding algorithm is illustrated in the flowchart in Figure 4.3.

Example 1: The binary and integer representation of inverse Gray code $\mathbf{v} = [1 \ 0 \ 1 \ 1]$ can be obtained as follows:

- Since the number of bits k is even, Initialize $b_0 = 0$.
- Since $v_1 \oplus v_2 \oplus v_3 \oplus v_4 = 1$ (denotes that integer h is odd), using the relationship $b_i = \overline{b_{i-1} \oplus v_i}$, we have,

$$b_1 = \overline{b_0 \oplus v_1} = \overline{0 \oplus 1} = 0$$

$$b_2 = \overline{b_1 \oplus v_2} = \overline{0 \oplus 0} = 1$$

$$b_3 = \overline{b_2 \oplus v_3} = \overline{1 \oplus 1} = 1$$

$$b_4 = \overline{b_3 \oplus v_4} = \overline{1 \oplus 1} = 1$$

Thus, $\mathbf{b} = [0 \ 1 \ 1 \ 1]$ or integer $h = 7$.

Example 2: The binary and integer representation of inverse Gray code $\mathbf{v} = [0 \ 0 \ 1]$ can be obtained as follows:

- Since the number of bits k is odd, Initialize $b_0 = 0 \oplus 0 \oplus 1 = 1$.

- Since $v_1 = 0$ (denotes that integer h is even), using the relationship $b_i = b_{i-1} \oplus v_i$, we have,

$$b_1 = b_0 \oplus v_1 = 1 \oplus 0 = 1$$

$$b_2 = b_1 \oplus v_2 = 1 \oplus 0 = 1$$

$$b_3 = b_2 \oplus v_3 = 1 \oplus 1 = 0$$

Thus, $\mathbf{b} = [1 \ 1 \ 0]$ or integer $h = 6$.

4.4.3 Distance Properties of Inverse Gray-coded Integers

Following the encoding algorithm in Section 4.4.1, the 3-bit, 4-bit and 5-bit inverse Gray codes can be derived as in Table 4.16.

It can be observed from Table 4.16 that Hamming distance of inverse Gray code for two consecutive integers h and $h \pm 1$ is at least $k - 1$, i.e. $d(\mathbf{v}_h, \mathbf{v}_{h \pm 1}) \geq k - 1$.

As an example, take the four-bit inverse Gray code in Table 4.14. The Hamming distance between any two inverse Gray codes corresponding to integers h_1 and h_2 ($0 \leq h_1, h_2 \leq 15$), i.e. $d(\mathbf{v}_{h_1}, \mathbf{v}_{h_2})$ are shown in Table 4.17.

Table 4.18 shows the list of neighbors of each integer h in a four-bit inverse Gray system, where $0 \leq h \leq 15$. Notice that none of the integers has neighbors that differ by ± 1 from itself.

The non-existence of integer(s) that differ by ± 1 as neighbors shows that undetectable errors due to neighboring residue vectors are eliminated, which results in an increase in the error detection probability. The following section describes the error detection of RSNS coded with inverse Gray, referred to as *inverse Gray RSNS* (IGRSNS) in the following discussions.

Figure 4.2: Encoding of inverse Gray codes

Figure 4.3: Decoding of inverse Gray codes

Integer h	Inverse Gray Codes $\mathbf{v}$		
	$k = 3$	$k = 4$	$k = 5$
0	000	0000	00000
1	110	1110	11110
2	011	0011	00011
3	101	1101	11101
4	010	0110	00110
5	100	1000	11000
6	001	0101	00101
7	111	1011	11011
8	—	1100	01100
9	—	0010	10010
10	—	1111	01111
11	—	0001	10001
12	—	1010	01010
13	—	0100	10100
14	—	1001	01001
15	—	0111	10111
16	—	—	01000
17	—	—	10110
18	—	—	01011
19	—	—	10101
20	—	—	01110
21	—	—	10000
22	—	—	01101
23	—	—	10011
24	—	—	00100
25	—	—	11010
26	—	—	00111
27	—	—	11001
28	—	—	00010
29	—	—	11100
30	—	—	00001
31	—	—	11111

Table 4.16: 3-bit, 4-bit and 5-bit inverse Gray codes

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	3	2	3	2	1	2	3	2	1	4	1	2	1	2	3
1	3	0	3	2	1	2	3	2	1	2	1	4	1	2	3	2
2	2	3	0	3	2	3	2	1	4	1	2	1	2	3	2	1
3	3	2	3	0	3	2	1	2	1	4	1	2	3	2	1	2
4	2	1	2	3	0	3	2	3	2	1	2	3	2	1	4	1
5	1	2	3	2	3	0	3	2	1	2	3	2	1	2	1	4
6	2	3	2	1	2	3	0	3	2	3	2	1	4	1	2	1
7	3	2	1	2	3	2	3	0	3	2	1	2	1	4	1	2
8	2	1	4	1	2	1	2	3	0	3	2	3	2	1	2	3
9	1	2	1	4	1	2	3	2	3	0	3	2	1	2	3	2
10	4	1	2	1	2	3	2	1	2	3	0	3	2	3	2	1
11	1	4	1	2	3	2	1	2	3	2	3	0	3	2	1	2
12	2	1	2	3	2	1	4	1	2	1	2	3	0	3	2	3
13	1	2	3	2	1	2	1	4	1	2	3	2	3	0	3	2
14	2	3	2	1	4	1	2	1	2	3	2	1	2	3	0	3
15	3	2	1	2	1	4	1	2	3	2	1	2	3	2	3	0

Table 4.17: Hamming distance d of four-bit inverse Gray code

Integer h	Neighbors			
0	5	9	11	13
1	4	8	10	12
2	7	9	11	15
3	6	8	10	14
4	1	9	13	15
5	0	8	12	14
6	3	11	13	15
7	2	10	12	14
8	1	3	5	13
9	0	2	4	12
10	1	3	7	15
11	0	2	6	14
12	1	5	7	9
13	0	4	6	8
14	3	5	7	11
15	2	4	6	10

Table 4.18: Neighbors of four-bit inverse Gray code

4.4.4 Error Detection using Inverse Gray RSNS

Similar with the previous error analysis of binary and Gray RSNS, error detection in inverse Gray RSNS can also be divided into three cases — single bit error in 1, 2 and 3 MRSs.

Case 1: Single bit error in one MRS

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *inverse Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *one* MRS, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.19.

$\mathbf{r}$	2													
	2													
	3													
h	0													

 $\rightarrow$

$\mathbf{r}'$	7	9	11	15	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	7	9	11	15	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	6	8	10	14		
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.19: Inverse Gray RSNS — Single bit error in *one* MRS for $\mathbf{r} = [2 \ 2 \ 3]$

It can be seen from Table 4.19 that for the given case, inverse Gray RSNS can detect all single bit errors in one MRS, i.e. $P_D = 1$.

Case 2: Single bit error in two MRSs

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *inverse Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *two* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.20.

All errors are successfully detected in the case of single bit error in two MRSs for the given case. Hence, $P_D = 1$.

$\mathbf{r}$	2																
	2																
	3																
h	0																

 $\rightarrow$

$\mathbf{r}'$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	7	9	11	15	7	9	11	15	7	9	11	15	7	9	11	15	15
	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.20: Inverse Gray RSNS — Single bit error in *two* MRSs for $\mathbf{r} = [2 \ 2 \ 3]$

Case 3: Single bit error in three MRSs

Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *inverse Gray* code ($k = 4$) is used in each residue channel, assuming single bit error in *all* MRSs, the possible permutations of received residue vectors $\mathbf{r}'$ are shown in Table 4.21.

$\mathbf{r}$	2																
	2																
	3																
h	0																

 $\rightarrow$

$\mathbf{r}'$	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

$\mathbf{r}'$	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
	7	7	7	7	9	9	9	9	11	11	11	11	15	15	15	15	15
	6	8	10	14	6	8	10	14	6	8	10	14	6	8	10	14	14
h'	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.21: Inverse Gray RSNS — Single bit error in *all* MRSs for $\mathbf{r} = [2 \ 2 \ 3]$

All errors can be detected in the case of single bit error in all MRSs for the given case. Hence, $P_D = 1$.

4.4.5 Numerical Results: Error Detection Probability of Binary, Gray and Inverse Gray RSNS

In the above sections, one example (i.e. transmission of residue vector $\mathbf{r} = [2 \ 2 \ 3]$ in RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$) has been given to show how inverse Gray RSNS can be used for error detection. It is seen that for the given case, inverse Gray RSNS is able to detect all single bit errors ($P_D = 1$) in any number of MRSs. Next, computer simulations are conducted to evaluate the error detection capability of inverse Gray RSNS, and its performance is compared with binary and Gray RSNS.

In this simulation, residue vectors $\mathbf{r}$ within dynamic range $\hat{M}$ are transmitted using k -bit binary, Gray and inverse Gray codes. For each residue vector transmitted, single bit noise perturbation is introduced in one, two and three MRSs respectively. Similarly, all permutations of possible received residue vector $\mathbf{r}'$ are considered. The validity of each residue vector is then verified by comparing $\mathbf{r}'$ with all LUT entries. If $\mathbf{r}'$ does not exist in the LUT, error detection is successful. Otherwise, undetectable error has occurred. Probability of error detection is computed by counting the number of successful error detection for each transmitted residue vector.

This exercise is repeated for all residue vectors within $\hat{M}$. Each individual probability computed earlier is then averaged across $\hat{M}$ to obtain the average error detection probability P_D for a given moduli set $\mathbf{m}$. This simulation is carried out for several moduli sets of the form $\mathbf{m} = [m-1 \ m \ m+1]$ and several values of bits per residue k (where $k \geq \lceil \log_2(m+2) \rceil$) for binary, Gray and inverse Gray codes respectively.

The complete numerical results are given in Table A.1 in Appendix A. For a more visual comparison, these results are also presented in graphical form in Figure A.1, A.2 and A.3 for $k = 4, 5$ and 6 respectively.

Figure 4.4 shows one subgraph for the comparison of error detection probability

for number of bits/residue $k = 6$.

Figure 4.4: Probability of detection in RSNS for $k = 6$

It can be seen that inverse Gray RSNS outperforms binary and Gray RSNS in terms of error detection probability. Due to the short dynamic range of RSNS and large Hamming distance between neighboring integers in inverse Gray, the error detection probability in inverse Gray RSNS is very high ($P_D > 0.9$) for any moduli set and number of bits per residue k used. Note that unlike RNS, RSNS has very good error detection performance without introducing any redundancy. The analysis and simulation studies in this direction confirm the existence of a powerful error detection code named *inverse Gray RSNS* (IGRSNS) code.

4.5 Error Correction in RSNS

In the previous section, numerical results have shown that RSNS has good inherent error detection capability, and its performance is further improved by inverse Gray codes that maximize the probability of detection P_D . Note that the actual detection probability P_D achieved is better than the expected theoretical performance $E[P_D]$. This implies that Inverse Gray code is an efficient binary representation that maximizes the utilization of holes for error detection. Since error correction can be performed only if error is successfully detected, good error detection is crucial as it determines the upper bound of correction ability.

Inverse Gray RSNS is undoubtedly a powerful class of *error detection code*, but it is incomplete without good error correction. In this section, the error correction feature of RSNS is explored. The following discussions assume single bit errors in single and multiple MRSs. Developing error correction algorithms for multi bit errors in MRSs are rather complex and have not been included within the scope of the present work.

4.5.1 Error Correction without Redundancy

Error correction in RSNS can be achieved without introducing any redundancy. If an error is detected in the received residue vector $\mathbf{r}'$, an intuitive method to correct error is by re-perturbing $\mathbf{r}'$ by single bit in one of its residues and listing out all possible permutations $\tilde{\mathbf{r}}$. If this perturbation results in *one* and only one valid residue vector, i.e. $n(\tilde{\mathbf{r}}) = 1$, this residue vector is taken as the corrected residue vector $\hat{\mathbf{r}}$. This perturbation is carried out due to the assumption that the *most frequent* type of error is single bit error in one MRS. This assumption is valid because the probability of occurrence of i errors is less than the probability of occurrence of j errors (for $i > j$) [4].

However, it is obvious that this method will not achieve a good error correction performance. Because the erroneous residue vector is only reperturbed by *single bit* in *one* of its residues, it is unable to correct "higher-degree" errors that occur within multiple MRSs. To achieve a satisfactory correction performance, redundancy is added by appending an additional MRS into the previously introduced N -moduli RSNS.

4.5.2 Introduction of Redundancy in RSNS

Redundancy can be added into RSNS to improve the error detection and correction performance. The more the number of redundant moduli introduced in RSNS, the better the correction ability. However, as shown in the later sections, appending *one* additional modulus m_{N+1} to RSNS moduli set $\mathbf{m} = [m_1 \ m_2 \ \dots \ m_N]$ is sufficient to correct most single bit errors.

The error correction ability depends on how the additional residue is designed. Ideally speaking, the *optimal* method to achieve a good error correction performance ($P_C \rightarrow 1$) is by constructing an MRS whose value is manually determined by exhaustive search of all residue vectors in the LUT. Different values of redundant residues must be assigned to the residue vectors that have high correlation between each other. However, this approach is not feasible in practice due to the amount of time needed to exhaustively take all pairs of residue vectors within $\hat{M}$, compute their correlation values, and decide the value of the redundant residue.

A *suboptimal* way of designing the redundant residue r_{N+1} is by using modulo arithmetic with respect to moduli m_{N+1} . Assuming three-moduli RSNS with moduli set $\mathbf{m} = [m-1 \ m \ m+1]$, the fourth residue (MRS) r_4 is formed by modulo

arithmetic of LUT index h with respect to moduli m_4 , where

$$m_4 = 2^k - 1 \quad (4.8)$$

and $k \geq \lceil \log_2(m+2) \rceil$ is the number of bits used in each residue channel. Since this fourth modulus is formed by modulo arithmetic, in the following discussion, this additional MRS is referred to as *checking residue*.

An example of three-moduli RSNS structure incorporating checking residue r_4 with moduli set $\mathbf{m} = [3 \ 4 \ 5]$ and number of bits $k = 4$ is shown in Table 4.22.

$r_1(m_1 = 3)$	2	2	3	3	3	2	2	2	1	1	1	0	0	0	1
$r_2(m_2 = 4)$	2	3	3	3	4	4	4	3	3	3	2	2	2	1	1
$r_3(m_3 = 5)$	3	3	3	2	2	2	1	1	1	0	0	0	1	1	1
$r_4(m_4 = 15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
h	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$r_1(m_1 = 3)$	1	1	2	2	2	3	3	3	2	2	2	1	1	1	
$r_2(m_2 = 4)$	1	1	0	0	1	1	1	2	2	2	3	3	3	4	
$r_3(m_3 = 5)$	2	2	2	3	3	3	4	4	4	5	5	5	4	4	
$r_4(m_4 = 15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
h	15	16	17	18	19	20	21	22	23	24	25	26	27	28	

$r_1(m_1 = 3)$	0	0	0	1	1	1	2	2	2	3	3	3	2	2	
$r_2(m_2 = 4)$	4	4	3	3	3	2	2	2	1	1	1	0	0	0	
$r_3(m_3 = 5)$	4	3	3	3	2	2	2	1	1	1	0	0	0	1	
$r_4(m_4 = 15)$	14	0	1	2	3	4	5	6	7	8	9	10	11	12	
h	29	30	31	32	33	34	35	36	37	38	39	40	41	42	

Table 4.22: LUT structure for moduli set $\mathbf{m} = [3 \ 4 \ 5]$ with checking residue r_4 and $k = 4$

4.5.3 Error Correction Algorithm

The proposed error correction algorithm in RSNS is as follows:

- Perturb the received residue vector $\mathbf{r}'$ by single bit in each residue and list out all the possible permutations $\tilde{\mathbf{r}}$. If this perturbation results in *one* valid residue vector, i.e. $n(\tilde{\mathbf{r}}) = 1$, it is taken as the corrected residue vector $\hat{\mathbf{r}}$.
- If the perturbation results in *more than one* valid residue vector, i.e. $n(\tilde{\mathbf{r}}) > 1$, the checking residue r_4 is used to determine which candidate(s) are valid. The

candidate with the same checking residue as the received residue vector $\mathbf{r}'$ is determined as the corrected residue vector $\hat{\mathbf{r}}$. If more than one candidate have the same checking residue as $\mathbf{r}'$, the first candidate is selected.

- If the perturbation results in *no* valid residue vector, i.e. $n(\tilde{\mathbf{r}}) = 0$, all candidate vectors $\tilde{\mathbf{r}}$ obtained from previous stages are discarded. The fact that the perturbation results in no valid residue vector shows that either single bit error in multiple MRSs or burst error has occurred. All vectors in the LUT having the same checking residue as the received checking residue r'_4 are determined to be the new candidate vectors $\tilde{\mathbf{r}}$. The candidate having the minimum distance with the received residue vector $\mathbf{r}'$ is determined as the corrected residue vector $\hat{\mathbf{r}}$. If more than one candidate have the same distance with $\mathbf{r}'$, the first candidate is selected.

This correction algorithm is illustrated in the flowchart in Figure 4.5.

For illustration on this error correction algorithm, the following sections discuss some examples using *binary* code for different cases such as single bit error in one, two and three MRSs respectively. For simplicity, correction using Gray and inverse Gray codes is not shown as they follow the same mechanism as binary code.

Example 1: Single bit, one MRS

Consider the same example as Case 1 in Section 4.2.3 on page 64. Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *one* MRS, there are twelve possible permutations of received residue vectors $\mathbf{r}'$ as listed in Table 4.6. Out of these twelve possible received vectors, four errors cannot be detected.

Now take an example where the received vector is $\mathbf{r}' = [6 \ 2 \ 3]$. In this case,

Figure 4.5: Error correction algorithm in RSNS

error is detected since $\mathbf{r}'$ is invalid. This received vector is perturbed by single bit in one of its MRSs and all permutations are listed out in Table 4.23.

$\mathbf{r}'$	$\begin{matrix} 6 \\ 2 \\ 3 \end{matrix}$	$\rightarrow$	$\tilde{\mathbf{r}}$	$\begin{matrix} 2 & 4 & 7 & 14 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{matrix}$	$\begin{matrix} 6 \\ 0 \\ 3 \end{matrix}$	$\begin{matrix} 6 \\ 3 \\ 3 \end{matrix}$	$\begin{matrix} 6 \\ 6 \\ 3 \end{matrix}$	$\begin{matrix} 6 \\ 10 \\ 3 \end{matrix}$	$\begin{matrix} 6 \\ 2 \\ 1 \end{matrix}$	$\begin{matrix} 6 \\ 2 \\ 2 \end{matrix}$	$\begin{matrix} 6 \\ 2 \\ 7 \end{matrix}$	$\begin{matrix} 6 \\ 2 \\ 11 \end{matrix}$	
h'	$\times$		$\tilde{h}$	$\begin{matrix} 0 & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{matrix}$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.23: Correction for $\mathbf{r}' = [6 \ 2 \ 3]$

It can be seen that out of 12 possible permutations of $\tilde{\mathbf{r}}$, there is only *one* valid candidate vector after perturbation, i.e. $\tilde{\mathbf{r}} = [2 \ 2 \ 3]$ corresponding to integer $\tilde{h} = 0$. This vector is then selected as the transmitted residue vector $\hat{\mathbf{r}}$. In this case, it is a correct reception since $\hat{\mathbf{r}} = \mathbf{r}$.

Example 2: Single bit, one MRS

Now take another example where the received vector is $\mathbf{r}' = [0 \ 2 \ 3]$. In this case, error is detected since $\mathbf{r}'$ is invalid. This received vector can be corrected as in Table 4.24.

$\mathbf{r}'$	0												
	2												
	3												
h'	$\times$	$\tilde{\mathbf{r}}$	1	2	4	8	0	0	0	0	0	0	0
			2	2	2	2	0	3	6	10	2	2	2
			3	3	3	3	3	3	3	3	1	2	7
		$\tilde{h}$	$\times$	0	$\times$	$\times$	$\times$	31	$\times$	$\times$	12	$\times$	$\times$

Table 4.24: Correction for $\mathbf{r}' = [0 \ 2 \ 3]$: Step 1

In this case, it can be seen that there are three candidate vectors, i.e. $\tilde{\mathbf{r}} = [2 \ 2 \ 3]$, $[0 \ 3 \ 3]$ and $[0 \ 2 \ 1]$ corresponding to integer $\tilde{h} = 0, 31$ and 12 respectively. Since $n(\tilde{\mathbf{r}}) \geq 1$, the candidate vector must be selected based on the value of checking residue r_4 . From the LUT in Table 4.22, it can be seen that the checking residue for $\mathbf{r} = [2 \ 2 \ 3]$ is $r_4 = 0$. Assuming correct reception of the checking residue, the received vector can further be corrected as in Table 4.25.

$\mathbf{r}'$	0												
	2												
	3												
	0												
h'	$\times$	$\tilde{\mathbf{r}}$	1	2	4	8	0	0	0	0	0	0	0
			2	2	2	2	0	3	6	10	2	2	2
			3	3	3	3	3	3	3	3	1	2	7
			$\times$	0	$\times$	$\times$	$\times$	1	$\times$	$\times$	12	$\times$	$\times$
		$\tilde{h}$	$\times$	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.25: Correction for $\mathbf{r}' = [0 \ 2 \ 3]$: Step 2

It can be observed from Table 4.25 that only one valid candidate vector remains as the other two vectors have different value of checking residue, i.e. $\tilde{\mathbf{r}} = [2 \ 2 \ 3]$ corresponding to integer $\tilde{h} = 0$. This candidate vector is then selected as the corrected residue vector $\hat{\mathbf{r}}$. In this case, the original residue vector also can be correctly recovered, i.e. $\hat{\mathbf{r}} = \mathbf{r}$.

Example 3: Single bit, two MRSs

Consider the same example as Case 2 in Section 4.2.3 on page 65. Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *two* MRSs, 6 out of 48 possible permutations of received residue vectors $\mathbf{r}'$ are undetectable errors as listed in Table 4.7. The other 42 residue vectors can be potentially recovered.

Now take the case where the received vector is $\mathbf{r}' = [2 \ 0 \ 7]$. In this case, error is detected since $\mathbf{r}'$ is invalid. This received vector can be corrected as in Table 4.26.

$\mathbf{r}'$	2												
	0												
	7												
h'	$\times$												

 $\rightarrow$

$\tilde{\mathbf{r}}$	0	3	6	10	2	2	2	2	2	2	2	2	2
	0	0	0	0	1	2	4	8	0	0	0	0	0
	7	7	7	7	7	7	7	7	3	5	6	15	
$\tilde{h}$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	18	$\times$	$\times$	$\times$	$\times$

Table 4.26: Correction for $\mathbf{r}' = [2 \ 0 \ 7]$

In this case, since the perturbation only results in *one* valid candidate vector, i.e. $\tilde{\mathbf{r}} = [2 \ 0 \ 3]$ corresponding to integer $\tilde{h} = 18$, this vector is then selected as the transmitted residue vector $\hat{\mathbf{r}}$. However, it can be seen that this is an *incorrect* decision as $\hat{\mathbf{r}} \neq \mathbf{r}$. This decision is incorrect because the received residue vector $\mathbf{r}'$ has a shorter distance to the selected residue vector $\hat{\mathbf{r}}$ ($d(\mathbf{r}', \hat{\mathbf{r}}) = 1$) compared to the transmitted residue vector $\mathbf{r}$ ($d(\mathbf{r}', \mathbf{r}) = 2$).

Example 4: Single bit, two MRSs

Now take the case where the received vector is $\mathbf{r}' = [2 \ 3 \ 2]$. In this case, error is detected since $\mathbf{r}'$ is invalid. This received vector can be corrected as in Table 4.27.

As observed from Table 4.27, single bit perturbations of received vector $\mathbf{r}'$ yields 3 possible candidates, i.e. $\tilde{\mathbf{r}} = [3 \ 3 \ 2]$, $[2 \ 2 \ 2]$ and $[2 \ 3 \ 3]$ corresponding to integer $\tilde{h} = 3, 35$ and 1 respectively. Next, the checking residues of these 3 candidates are

Example 5: Single bit, three MRSs

Now, consider the same example as Case 3 in Section 4.2.3 on page 65. Consider a three-moduli RSNS with moduli set $\mathbf{m} = [3 \ 4 \ 5]$. If residue vector $\mathbf{r} = [2 \ 2 \ 3]$ corresponding to integer $h = 0$ is transmitted and four-bit *binary* code ($k = 4$) is used in each residue channel, assuming single bit error in *three* MRSs, only 1 out of 64 possible permutations of received residue vectors $\mathbf{r}'$ is an undetectable error as listed in Table 4.8 and the remaining 63 are potentially recoverable.

Assume that the received vector is $\mathbf{r}' = [0 \ 0 \ 7]$. In this case, error is detected since $\mathbf{r}'$ is invalid. This received vector can be corrected as in Table 4.30.

$\mathbf{r}'$	0											
	0											
	7											
h'	$\times$											

 $\rightarrow$

$\tilde{\mathbf{r}}$	1	2	4	8	0	0	0	0	0	0	0	0
	0	0	0	0	1	2	4	8	0	0	0	0
	7	7	7	7	7	7	7	7	3	5	6	15
$\tilde{h}$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 4.30: Correction for $\mathbf{r}' = [0 \ 0 \ 7]$: Step 1

As observed from Table 4.30, the single bit perturbations do not result in any valid vector. Thus, assuming correct reception of the checking residue, i.e. $r'_4 = r_4 = 0$, all residue vectors having $r_4 = 0$ are $\tilde{\mathbf{r}} = [2 \ 2 \ 3]$, $[1 \ 1 \ 2]$ and $[0 \ 4 \ 3]$ corresponding to integer 0, 15 and 30 respectively. The distance between each candidate vector $\tilde{\mathbf{r}}$ and $\mathbf{r}'$ is then measured and the candidate with the minimum distance is selected. This process is shown in Table 4.31.

$\mathbf{r}'$	0			
	0			
	7			
h'	$\times$			

 $\rightarrow$

$\tilde{\mathbf{r}}$	2	1	0
	2	1	4
	3	2	3
$d(\tilde{\mathbf{r}}, \mathbf{r}')$	3	4	2
$\tilde{h}$	0	15	30

Table 4.31: Correction for $\mathbf{r}' = [0 \ 0 \ 7]$: Step 2

Since vector $\tilde{\mathbf{r}} = [0 \ 4 \ 3]$ has the minimum distance with the received vector $\mathbf{r}'$ ($d(\tilde{\mathbf{r}}, \mathbf{r}') = 2$), it is determined as the corrected vector $\hat{\mathbf{r}}$. In this case, decision made is incorrect, since $\hat{\mathbf{r}} \neq \mathbf{r}$.

4.5.4 Numerical Results: Average Error Correction Probability of Binary, Gray and Inverse Gray RSNS

In the above sections, several examples on the error correction mechanism in RSNS have been demonstrated. Even though some correction attempts are successful, it is unable to claim that it is always able to correct any types of errors. In the above examples, the number of correction failures is quite high due to the use of *binary* code. For *inverse Gray* code, the correction capability of RSNS increases substantially due to the low probability of undetected errors. In the following section, computer simulations are conducted to evaluate the error correction capability of binary, Gray and inverse Gray codes in RSNS.

In this simulation, residue vectors $\mathbf{r}$ within dynamic range $\hat{M}$ are transmitted using k -bit binary, Gray and inverse Gray codes. For each residue vector transmitted, single bit noise perturbation is introduced in one, two and three MRSs respectively. All permutations of possible received residue vector $\mathbf{r}'$ are considered. The validity of each residue vector is then verified by comparing $\mathbf{r}'$ with all LUT entries. If $\mathbf{r}'$ does not exist in the LUT, error detection is successful. Otherwise, if it exists, undetectable error has occurred.

For each successful error detection, error correction is performed using the algorithm described in Figure 4.5. If the corrected residue vector $\hat{\mathbf{r}}$ is similar with the transmitted residue vector $\mathbf{r}$, error correction is successful. Otherwise, if $\hat{\mathbf{r}} \neq \mathbf{r}$, error correction fails. The probability of error correction is then computed by counting the number of successful error correction for each transmitted residue vector.

This exercise is repeated for all residue vectors within $\hat{M}$. Each individual probability computed earlier is then averaged across $\hat{M}$ to obtain the average error correction probability P_D for a given moduli set $\mathbf{m}$. This simulation is carried out for several moduli sets of the form $\mathbf{m} = [m-1 \ m \ m+1]$ and several values of bits

per residue k (where $k \geq \lceil \log_2(m+2) \rceil$) for binary, Gray and inverse Gray codes respectively.

The complete numerical results are given in Table B.1 in Appendix B. For a more visual comparison, these results are also presented in graphical form in Figure B.1, B.2 and B.3 for $k = 4, 5$ and 6 respectively.

Figure 4.6 shows one subgraph for the comparison of error correction probability for number of bits/residue $k = 6$.

Figure 4.6: Probability of correction in RSNS for $k = 6$

It can be seen that inverse Gray RSNS also outperforms binary and Gray RSNS in terms of error correction probability for all cases of single bit errors. This is a result of the high probability of detection P_D and the large Hamming distance between consecutive integers in inverse Gray RSNS. Besides, unlike RNS which can correct errors only when at least two redundant moduli are present, RSNS can correct errors with good performance with only *one* additional modulus.

4.6 Advantages of Inverse Gray RSNS (IGRSNS) over RNS

From the above discussions, it is obvious that IGRSNS outperforms RNS (and RRNS) in many aspects, such as:

- Number of redundant moduli

In RNS, the addition of R redundant moduli makes it able to detect R errors and correct $\lfloor \frac{R}{2} \rfloor$ errors. In other words, to *detect* one erroneous residue, it needs at least one redundant modulus; and to *correct* one erroneous residue, it needs at least two redundant moduli. However, in IGRSNS, error detection is achieved without any redundancy and it is able to detect errors not only in one erroneous residue, but also when *two* or even *all* MRSs are in error. Besides, IGRSNS is also able to achieve a reasonable error correction performance in *any* number of MRSs by the addition of only *one* redundant modulus. In RNS, to achieve correction in three erroneous moduli, at least six redundant moduli are needed. This makes RSNS very attractive for replacing RNS in the application of error control.

- Computational complexity

In RSNS, since LUT is used, computational complexity is relatively lower compared to RNS which mainly uses CRT for decoding. This is achieved at the expense of decoding time needed due to the exhaustive comparison of residue vector with each LUT entry. However, the size of LUT is much smaller due to the smaller dynamic range of RSNS.

4.7 Summary

In this chapter, the application of RSNS for error control has been introduced. Several binary representation schemes, such as conventional binary, Gray and inverse Gray codes have been introduced for RSNS. From the theoretical analysis and numerical results, it is obvious that inverse Gray code outperforms binary and Gray codes in terms of error detection and correction capability in RSNS. This result confirms the existence of a new class of powerful binary representation, namely the *inverse Gray RSNS* (IGRSNS) for the application in error control. Besides, the good detection and correction performance makes IGRSNS potential to be used for some other applications, such as in digital arithmetic, multi-precision computing, digital signal processing (for example, design of fast convolution circuits and filters) and communication systems. In this context, further application of IGRSNS into communication systems is considered and is presented in the next chapter.

Chapter 5

Application of IGRSNS in Digital Communication Systems

In the previous chapter, the application of IGRSNS for error control has been discussed. The robust performance due to the good error detection and correction capability makes IGRSNS potential to be applied into other practical applications such as fault tolerant computing and communication systems. However, the present work focuses the applications of RSNS in communication related areas. In this context, the application of RSNS in Code Division Multiple Access (CDMA) systems is considered.

In an N -moduli IGRSNS-CDMA, any bit string with length b representing information symbol h can be uniquely and unambiguously represented as a set of inverse Gray coded residues $[v_1 \ v_2 \ \dots \ v_N]$ with respect to PRP moduli set $[m_1 \ m_2 \ \dots \ m_N]$, where v_i represents the inverse Gray code of the RSNS residue of h with respect to m_i for $i = 1, \dots, N$. The RSNS dynamic range $\hat{M}$ determines the maximum number of bits possible in a symbol b , where $b \leq \lfloor \log_2 \hat{M} \rfloor$. Also, a checking residue r_{N+1} is used to improve the error correction performance of IGRSNS in CDMA systems.

5.1 Transceiver Structure of IGRSNS-CDMA

5.1.1 Pre-transmission: Construction of LUT

At the pre-transmission stage, a lookup table (LUT) needs to be constructed at the encoder and decoder unit of the transmitter and receiver. First, a moduli set is selected based on the system requirement. For the selection of moduli values, the following constraints need to be considered:

- The RSNS dynamic range $\hat{M}$ of the moduli set should be higher than 2^b , where b is the number of bits per symbol.
- The difference between dynamic range $\hat{M}$ and 2^b should be kept as low as possible, since dynamic range determines the length of the LUT, and thus the receiver complexity.

Suppose that the system is required to represent Z unique integers (where $Z = 2^b$ and b is the number of bits per symbol), the moduli set is first selected by equation (3.22). Since the longest sequence is located from the *start* position in RSNS, all integers within the range $[start, start + Z)$ are then encoded by equation (3.4) and stored into the LUT. This LUT is then used for encoding and decoding at the transmitter and receiver.

5.1.2 Transmitter Structure

Figure 5.1 shows the transmitter block diagram of the proposed IGRSNS-CDMA system.

The IGRSNS encoder first converts the M -ary symbols into parallel residue channels. Conventionally, in RNS-CDMA, these residue values are *mapped* into orthogonal sequences and transmitted in parallel. In such cases, the total number

Figure 5.1: Block diagram of IGRSNS-CDMA transmitter

of orthogonal sequences required is the sum of all moduli values, i.e. $\sum_{i=1}^N m_i$. This thesis suggests a method to reduce the required number of orthogonal sequences without affecting the performance of the system. Here, each residue value is independently modulated and spread. In such context, the number of orthogonal spread codes required is equal to the number of moduli N , which is significantly lesser compared to the ones in conventional RNS-CDMA system.

For analysis, this work assumes the use of three-moduli IGRSNS with moduli set of the form $[m-1 \ m \ m+1]$ and one checking residue for the proposed system. The result of this research can be extended to a general three-moduli RSNS with any form of moduli as well as N -moduli RSNS. In the following sections, each block of the transmitter is described in greater detail.

IGRSNS Encoder

At the transmitter, the M -ary symbol h is encoded into integer Gray representation by the use of an IGRSNS encoder. First, integer h is used as an index to select the LUT entry. The selected LUT entry contains the RSNS representation $[r_1 \ r_2 \ r_3]$ corresponding to integer h . This residue vector is then encoded into its integer Gray representation $[v_1 \ v_2 \ v_3]$. At the same time, the value of the checking residue is also computed by modulo arithmetic with respect to the fourth modulus m_4 , where

$m_4 = 2^k - 1$ and k is the number of bits per residue used in each residue channel.

Modulation

The integer Gray residues generated by the IGRSNS encoder are independently modulated using a uniform modulation scheme. The modulation scheme used depends on the maximum value of the modulus. If moduli set of form $[m-1 \ m \ m+1]$ is used, the modulation scheme depends on the value of $m+1$. Since the smallest PRP moduli set possible for moduli set of this form is $\mathbf{m} = [3 \ 4 \ 5]$, the smallest modulation scheme that can be used in RSNS is 8-PSK (Phase Shift Keying). To simplify implementation, Quadrature Amplitude Modulation (QAM) is used in this proposal. 16-QAM and 64-QAM are used depending on the moduli set values.

The constellation diagram of 16-QAM and 64-QAM is presented in Figure 5.2(a) and 5.2(b) respectively. The integer value in the parenthesis shows the equivalent integer h in inverse Gray representation.

As can be observed from Figure 5.2, consecutive integers are located far apart in the constellation. This is the effect of inverse Gray representation which maximizes Hamming distance between consecutive integers.

Table 5.1 shows recommended moduli sets $\mathbf{m}$ for several values of bits per symbol b . The moduli set determines the number of bits per residue k , the code rate as well as the corresponding modulation type.

Bits/symbol b	Dynamic Range	Moduli Set $\mathbf{m}$	Bits/residue k	Code Rate	Modulation
4	16	[3 4 5]	4	1/3	16-QAM
6	64	[5 6 7]	4	1/2	16-QAM
8	256	[11 12 13]	4	2/3	16-QAM
10	1024	[25 26 27]	6	5/9	64-QAM
12	4096	[51 52 53]	6	2/3	64-QAM

Table 5.1: Recommended moduli sets and modulation types for several values of bits per symbol b

Figure 5.2: (a) Constellation for 16-QAM modulation scheme (b) Constellation for 64-QAM modulation scheme. Integer value in the parenthesis shows the equivalent integer h of the corresponding binary value in inverse Gray representation.

Spreading

The modulated output from each residue channel is individually spread using orthogonal spread codes. There are various candidates for spread codes, such as Walsh-Hadamard codes, pseudo-noise (PN) sequence, M -sequence, Gold codes and Kasami sequence. In this thesis, Walsh-Hadamard codes are recommended due to the ease of generation and perfect orthogonality.

In contrast with the RNS-CDMA proposed in Chapter 2 whose number of spread codes needed is the sum of all moduli values, the number of spread codes in the proposed system is equal to the number of moduli N . For example, for three-moduli RSNS with one redundant modulus, *only* four orthogonal spread codes are required for each user. These codes are then used for spreading the data, and the spread data from each residue channel are combined together and transmitted after appropriate pulse shaping.

5.1.3 Receiver Structure

The receiver block diagram for the proposed IGRSNS-CDMA system is shown in Figure 5.3(a).

Similar to other number theoretic transforms (NTT), the residue operations belonging to different moduli in IGRSNS are mutually independent [3] [4]. In line with this property, the receiver is partitioned into a number of independent parallel receivers, each dedicated to receiving one residue digit. The total number of such parallel receivers is equal to the total number of residue channels. Each residue channel should be able to retrieve the corresponding data. Details of each sub-receiver unit is shown in Figure 5.3(b).

Figure 5.3: (a) Generalized block diagram of IGRSNS-CDMA Receiver (b) Details of subreceiver for i^{th} residue channel in Figure 5.3(a). (h is the channel information and P is the number of paths)

Equalization

The subreceiver unit has coherent RAKE structures with a diversity order equal to the number of paths P . Assuming perfect synchronization, each subreceiver is multiplied with the conjugate of channel values h_i^* . The resulting outputs are then maximally ratio combined with their multipath replicas.

Despreading and Demodulation

The output from the equalizer is then despread with each of the orthogonal sequences belonging to the respective residue values. The despread data is then demodulated to get the estimated inverse Gray representation of the residue value v_i' of the corresponding residue channel.

IGRSNS Decoder

The estimated inverse Gray representation of the residue value from each residue channel $[v'_1 \ v'_2 \ v'_3]$ and the checking residue r'_4 are passed to IGRSNS decoder. First, the inverse Gray residues are converted into integers and passed to an error correction/erasure module. The validity of each residue vector is then verified by comparing it with each residue vector in the LUT. If the residue vector is valid, it is determined as the transmitted residue vector $\hat{\mathbf{r}}$. Otherwise, error correction is performed and each symbol is reconstituted using the appropriate residue channels after suitable error correction methods using checking residue r_4 . The LUT index corresponding to the corrected residue vector $\hat{\mathbf{r}}$ is determined as the transmitted M -ary symbol h' .

5.2 Simulation Results

The BER performance of the proposed system is evaluated using extensive computer simulations. The system assumes a constant chip rate, which is fixed at 3.84 Mcps. Since it uses orthogonal modulation and a constant chip rate, the spreading factor is an important design parameter in determining the symbol rate (and thereby the data rate). Computer simulations were carried out for several moduli sets $\mathbf{m} = [m-1 \ m \ m+1]$ as well as values of bits per residue k (where $k \geq \lceil \log_2(m+2) \rceil$). The details of the moduli sets used in the simulation as well as the proposed modulation scheme for these moduli sets are explained in Table 5.1.

For spreading, orthogonal Walsh sequences are used. In this case, since we are using three-moduli system with one redundant residue, the minimum spreading factor (SF) required is *four*. The present simulation system uses $\text{SF} = 4$, unless otherwise stated. The simulation was first carried out in Additive White Gaussian

Noise (AWGN) channel, which was then extended into multipath fading channel. No external coding methods are employed for the data bits. The system assumes perfect synchronization and perfect channel knowledge.

In the following section, several simulations are conducted to study the error performance of IGRSNS-CDMA. First, the BER performance of IGRSNS-CDMA is evaluated against RNS-CDMA, both in the case *before* and *after* redundancy is introduced. Next, the effect of the moduli set selection on the error performance of IGRSNS-CDMA is discussed. Finally, the performance of IGRSNS-CDMA in multipath fading channel is presented.

5.2.1 Performance Comparison of IGRSNS-CDMA and RNS-CDMA

The performance of the proposed IGRSNS-CDMA system is compared with the conventional RNS-CDMA. For a fair comparison, the RNS-CDMA transceiver structure is also modified such that it is equivalent to the one of IGRSNS-CDMA in Figure 5.1 and 5.3. The modified transceiver structure for RNS-CDMA with one redundant modulus is shown in Figure 5.4.

At the transmitter, the RNS encoder converts the M -ary symbols into parallel residues by modulo arithmetic. Each residue is independently modulated and spread by orthogonal sequences. The spread data from each residue channel are then combined together and transmitted. At the receiver, the noise perturbed signal is equalized with the available channel information. The equalized data is then independently despread by respective orthogonal sequence in each residue channel and demodulated. The residue values after demodulation are converted into M -ary symbols by standard methods such as CRT or mixed radix conversion (MRC) [24] [28].

Figure 5.4: Modified transceiver structure of RNS-CDMA (The subreceiver at the receiver is similar to the one in Figure 5.3(b).)

BER Performance without Redundancy

First, the performance of IGRSNS-CDMA is compared with RNS-CDMA without using any redundant modulus. Figure 5.5 shows the BER performance of IGRSNS-CDMA and RNS-CDMA for several moduli sets with the same data modulation in an AWGN channel.

It can be seen that without redundant modulus, the performance difference of IGRSNS-CDMA and RNS-CDMA is insignificant, although it can be noted that IGRSNS-CDMA performs *slightly* better (by about 1-2 dB). The BER performance is unsatisfactory in both cases due to transmission errors. As we know, without redundancy, RNS-CDMA is unable to perform error correction; whereas for IGRSNS-CDMA, error correction properties are very limited without redundant modulus. Thus, though many errors can be successfully detected, only few of them can be corrected for the proposed system.

Figure 5.5: Performance comparison of IGRSNS-CDMA and RNS-CDMA (without redundancy)

BER Performance with One Redundant Modulus

Next, the performance of IGRSNS-CDMA is evaluated against RNS-CDMA with the use of one redundant modulus. From equation (4.8), the checking residue of the IGRSNS-CDMA is formed by modulo arithmetic with respect to the fourth modulus m_4 , where $m_4 = 2^k - 1$ and k is the number of bits in each residue channel. Figure 5.6 compares the BER performance of IGRSNS-CDMA and RNS-CDMA for several moduli sets with the same data modulation in an AWGN channel.

It can be seen that with one redundant modulus, IGRSNS-CDMA outperforms RNS-CDMA in *all* cases. In fact, the performance of RNS-CDMA after the addition of redundant modulus is similar to the one without redundancy. This is due to the inability of RNS to perform error correction with only one redundant modulus. Recall from Chapter 2 that RNS can only correct errors when at least two redundant moduli are present, and the correction ability depends on the number of redun-

Figure 5.6: Performance comparison of IGRSNS-CDMA and RNS-CDMA (with one redundant modulus)

dant moduli. For correction of errors in one, two and three non-redundant moduli, RNS needs two, four and six redundant moduli respectively. However, in IGRSNS-CDMA, the presence of *one* redundant modulus is sufficient to correct most single bit errors in *any* number of MRSs. Therefore, having one redundant modulus is sufficient for IGRSNS unless when better BER performance is demanded under special circumstances.

On average, IGRSNS-CDMA improves the coding gain from RNS-CDMA by about 3-6 dB for any moduli sets used. This is due to the good inherent property of IGRSNS-CDMA which is further supported by the use of an error correction module.

5.2.2 Effect of Moduli Set Selection on BER Performance

In the previous sections, it has been seen that the selected moduli set determines the LUT length, and thus the receiver complexity. However, different moduli sets shows the use of different modulation type and different constellation diagram. Hence, the selection of moduli sets directly has an impact on BER performance. Figure 5.7 compares the performance of IGRSNS-CDMA for several moduli sets (with one checking residue) in an AWGN channel.

Figure 5.7: Effect of moduli sets on BER performance

In general, it can be observed that the BER performance of IGRSNS-CDMA for a particular modulation scheme degrades as the moduli values increase. This is due to the existence of more *valid* symbols in the constellation diagram, resulting in the decrease in the number of holes. This problem can be alleviated by increasing the number of bits per residue k , but this translates to higher modulation scheme which brings down the error tolerance further. This analysis ascertains our previous observation that the difference between dynamic range $\hat{M}$ and 2^b (where b is the

number of bits per symbol) must be kept as low as possible.

5.2.3 Performance of IGRSNS-CDMA in Multipath Channel

To make the simulation study complete, the performance of IGRSNS-CDMA is evaluated under realistic multipath channel. The simulation system uses moduli set $\mathbf{m} = [3 \ 4 \ 5]$ with one redundant modulus. As listed in Table 5.1, this moduli set uses $k = 4$ bits/residue, and the minimum modulation type required is 16-QAM. The value of the redundant modulus is determined by equation (4.8), i.e. $m_4 = 2^4 - 1 = 15$. For spreading, the simulation system uses Walsh code with a spreading factor (SF) of 64. For this analysis, a longer PG is selected to minimize the effect of the multipath interference caused by the delay spread introduced by the channel.

The simulation was carried out under AWGN and slow fading Rayleigh channel. A multipath channel model with exponential delay profile is considered. The multipath intensity profile is selected such that the amplitude of the third path is ten times smaller compared to the first path. The system assumes perfect synchronization and perfect channel knowledge. No other external channel coding methods is employed for the data bits. Power control methods are not considered. Figure 5.8 compares the performance of IGRSNS-CDMA for AWGN and Rayleigh channel.

It can be observed from Figure 5.8 that the BER performance in AWGN is independent of the spreading factor used. Also, the BER performance under multipath channel with low mobility degrades from that of AWGN by about 3 dB at $\text{BER} = 10^{-3}$ due to multipath interference. The degradation will be more significant for systems with high mobility, mainly due to errors in channel estimation. However, we have not considered the channel estimation and the related issues within the scope of the present thesis.

Figure 5.8: BER performance of IGRSNS-CDMA under AWGN and Rayleigh channels

5.3 Summary

Inverse Gray RSNS (IGRSNS) have good inherent error detection and correction ability due to the short dynamic range and the large Hamming distance between neighboring integers. In this chapter, the application of IGRSNS in CDMA systems was demonstrated. When one checking residue is incorporated, the proposed system is able to achieve a coding gain of 3-6 dB as compared to the conventional RNS-CDMA systems. The bandwidth efficiency, reduction in computational complexity due to the use of LUT as well as robustness against channel impairments make the proposed system a good candidate for high-speed data transmission as well as next generations of broadband communication systems.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

Number Theoretic Transforms (NTT) have some very attractive properties, such as carry free arithmetic and lack of ordered significance among residue digits [3] [4]. These properties make NTTs attractive for implementation into a wide range of practical applications, such as digital processing, fault-tolerant computing, error control and communication systems. In this thesis, one subclass of NTT, namely the *Robust Symmetrical Number System* (RSNS), has been presented.

Conventional method of encoding and decoding in RSNS by means of Chinese Remainder Theorem (CRT) is of several order more complex compared to the equivalent Residue Number System (RNS). This makes it not feasible to be implemented in practice due to the hardware complexity of CRT operation. This problem can be alleviated by the use of lookup table (LUT) at the encoder and decoder. Although the use of LUT is generally slower, the use of LUT in RSNS is justifiable due to the short dynamic range.

The short dynamic range of RSNS also makes it more challenging for exploiting inherent error detection features without any redundancy as in RNS. Besides, it

was seen that due to the integer Gray property of RSNS sequence, the error detection performance is not optimal. A novel binary representation scheme, namely the *inverse Gray code*, is proposed to optimize the performance of error detection in RSNS. It was shown by extensive simulation studies that inverse Gray code outperforms the conventional binary code as well as Gray code in the context of RSNS. The resulting code, referred to as *inverse Gray RSNS* (IGRSNS), has a near-optimal error detection ability ($P_D \approx 1$) for any moduli sets. This confirms IGRSNS as a new class of powerful error detection code.

Besides detection, IGRSNS can also be used for error correction. However, it was shown that the performance of IGRSNS and RNS does not differ much if no redundant modulus is added. With *one* redundant modulus, the error correction of IGRSNS is satisfactory for single bit errors in any number of moduli. This cannot be achieved in RNS, since it needs at least two redundant moduli to perform error correction on only *one* modulus. For correction on three erroneous residues, RNS would require six redundant moduli, which is more than the number of non-redundant moduli itself. This translates to a much lower code rate compared to the equivalent IGRSNS, which only needs one redundant modulus to perform error correction.

Finally, the application of IGRSNS in communication systems (CDMA) was demonstrated. The transceiver structure for three moduli IGRSNS-CDMA with one redundant modulus was proposed. Simulation studies have shown that the proposed system outperforms RNS-CDMA in all cases in terms of BER performance. Besides, the effect of the selection of moduli sets on the BER performance has been studied. Some studies on the performance of IGRSNS-CDMA in AWGN as well as realistic multipath channel have also been conducted. The bandwidth efficiency, reduction in computational complexity due to the use of LUT and robustness

against channel impairments make the proposed system a good candidate for high-speed data transmission as well as next generations of broadband communication systems.

6.2 Future Works

Since RSNS is a new class of number theoretic transform, not much research has been conducted so far. The discussion of this number system in the available literature is also limited.

RSNS has a lot of attractive features and potential that can be further explored. Some of the possible areas of research for future works include:

Improvement of Error Correction Algorithm

RSNS has a good inherent error detection ability due to its short dynamic range and the large number of holes available. This error detection performance is improved and optimized by the inverse Gray code that combats the integer Gray coding property of RSNS and encourages error-perturbed residue vectors to fall into holes. The excellent error detection performance of RSNS convinces one that it can be used for efficient error correction.

In this thesis, an error correction algorithm has been proposed in Section 4.5. It was shown that the proposed algorithm can achieve a reasonably good performance with one redundant modulus. However, this performance can be improved further by other alternative error correction mechanisms. In the proposed algorithm, in the event when an erroneous residue vector $\mathbf{r}' : \mathbf{r}' \notin \text{LUT}$ is received, it is first perturbed by single bit in each residue and all the permutations are listed. This step is carried out due to the assumption that the most frequent type of error is single bit errors. Although this assumption is correct in general, it is not always

practical to ignore the presence of multi-bit errors in multiple MRSs. Thus, the present studies of carrying out single bit perturbation poses an inherent limit on the error correction ability. The error correction algorithms for multi-bit scenarios are much more complex and need to be studied in detail for the widespread practical usage of RSNS for error control applications.

Performance Improvement in Communication Systems Applications

In this thesis, the use of IGRSNS for applications in communication systems has been considered. Simulation studies have shown that the proposed IGRSNS-CDMA system outperforms RNS-CDMA in all cases in terms of BER performance. Nevertheless, the performance can still be improved by using a better correction algorithm.

One good alternative for correction algorithm that can improve performance in communication systems is by using a context-specific information, such as *soft-decision decoding*. So far, the present error correction methods for IGRSNS have been focused on hard-decision decoding only. Soft-decision decoding based on some distance measures such as Euclidean distance can be exploited in this context. Similar to other codes that use soft-decision decoding such as the Soft Output Viterbi Algorithm (SOVA) or Turbo codes, it is expected that performance of IGRSNS will substantially improve if soft-decision decoding is used.

Besides, the performance under realistic multipath environment should be analyzed by relaxing the assumptions on channel estimations and synchronization. The effect of errors in channel estimation and synchronization is much more significant in performance for a multipath environment. Efficient channel estimation and synchronization algorithms need to be explored. For example, adaptive semi-blind channel estimation method with specialized pilot symbols [40] [41] is one of the

possible research areas in this context.

Since IGRSNS (as well as RNS) converts an integer into *parallel* residues, it appears as if there are *multiple* users in the system. Thus, the problem of multiple access interference (MAI) comes into picture. Some interference reduction methods such as the one proposed by Verdu in [42] can be used to mitigate MAI. The present simulation results did not assume the use of any interference reduction methods or other advanced signal processing algorithms for performance improvement.

Studies can also be extended into other realistic channel environments, such as Nakagami and Rician multipath channel models. Since BER performance highly depends on the channel behaviour, statistical characteristics of these channel models (including Rayleigh) should be analyzed further. Better coding schemes can be devised to suit the behavior of these channel models and better error correction mechanisms can be explored for future improvement in communication systems applications.

Performance Analysis under Burst Errors

So far, the studies conducted have been focused on the performance evaluation of IGRSNS under *single bit* errors in one or more moduli. This is justifiable as the probability of occurrence for single bit errors is higher compared to burst errors in realistic communication channels. It has also been shown that inverse Gray RSNS outperforms conventional binary RSNS, Gray RSNS and RNS in terms of robustness against single bit errors. This performance analysis can be further extended for burst errors and other timing jitters which are very frequent in practical communication systems. More theoretical formulation and statistical analysis as well as simulation studies for burst errors and timing jitters are required in this context.

Application of IGRSNS in Fault-tolerant Architectures and Digital Circuit Design

Similar with RNS, IGRSNS has a potential to be applied not only in the field of error control and communication systems, but also in other practical applications such as fault-tolerant architectures and digital signal processing. Due to the carry free arithmetic and the ability to isolate individual digits, IGRSNS is a leading candidate for use in digital signal processing, such as design of filters and convolutional processors. Also, IGRSNS facilitates the realization of low-power arithmetic, which is critical in many current systems and especially in embedded processors [3].

Fault-tolerance is an area that was subject of much research in the early days of computing, but it subsequently declined when computer technology became more reliable. Now, with the advent of extremely dense computer chips that cannot be fully tested, fault tolerance and the general area of computational integrity have again become more important [43]. Besides, VLSI design for these applications will be a very promising and attractive area for the continuation of this work.

Appendix A

Error Detection Ability of Binary, Gray and Inverse Gray Codes

Figure A.1, A.2 and A.3 compare the error detection capability of binary, Gray and inverse Gray codes under single bit noise perturbation in one, two and three MRSs. The simulation was carried out for several moduli sets of the form $\mathbf{m} = [m-1 \ m \ m+1]$ and several values of bits per residue k (where $k \geq \lceil \log_2(m+2) \rceil$). Subsequently, the complete numerical results are presented in Table A.1.

It can be observed that in general, inverse Gray RSNS (IGRSNS) outperforms binary and Gray RSNS in terms of error detection probability. Due to the short dynamic range of RSNS and long Hamming distance of neighboring integers in inverse Gray, the error detection probability in IGRSNS is very high ($P_D > 0.9$) for any moduli set and number of bits per residue k . IGRSNS is able to demonstrate good error detection performance without any redundancy.

(a) Number of bits per residue $k = 4$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[13 14 15]	84.22	78.45	95.90	87.41	87.83	92.39	89.59	85.13	87.74
[9 10 11]	83.84	77.89	96.34	89.77	88.71	95.73	92.73	88.19	95.33
[7 8 9]	80.83	77.44	96.74	88.28	88.41	98.28	91.24	88.06	98.73
[3 4 5]	76.74	75.58	98.06	89.83	89.05	99.81	94.19	91.64	100

(b) Number of bits per residue $k = 5$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[29 30 31]	89.30	83.90	97.60	90.43	92.14	95.2	93.59	90.83	91.72
[23 24 25]	88.80	83.50	97.49	92.07	92	96.48	94.58	91.79	96.9
[17 18 19]	88.45	83.17	98.15	92.39	92.53	99.27	95.14	92.52	99.81
[13 14 15]	87.37	82.76	100	91.94	92.21	100	94.67	92.38	100
[9 10 11]	87.07	82.31	100	93.46	92.78	100	96.28	93.95	100
[7 8 9]	84.66	81.95	100	92.5	92.58	100	95.51	93.89	100
[3 4 5]	81.40	80.47	100	93.49	92.99	100	97.02	95.72	100

(c) Number of bits per residue $k = 6$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[61 62 63]	92.28	87.36	99.36	92.5	94.68	97.72	95.85	94.11	96.63
[51 52 53]	92.07	87.20	99.45	93.92	94.69	98.59	96.5	94.97	97.32
[41 42 43]	92.08	86.97	99.62	94.06	94.87	99.17	96.9	95.04	98.74
[29 30 31]	91.09	86.59	99.78	93.35	94.54	99.87	96.29	94.69	99.95
[23 24 25]	90.67	86.25	99.91	94.49	94.44	99.94	96.86	95.25	99.96
[17 18 19]	90.38	85.97	99.86	94.72	94.81	99.98	97.19	95.67	100
[13 14 15]	89.48	85.63	100	94.4	94.59	100	96.91	95.59	100
[9 10 11]	89.23	85.26	100	95.46	94.98	100	97.85	96.5	100
[7 8 9]	87.22	84.96	100	94.79	94.85	100	97.4	96.46	100
[3 4 5]	84.50	83.72	100	95.48	95.13	100	98.28	97.52	100

Table A.1: Probability of error detection P_D in RSNS using binary, Gray and inverse Gray (IG) codes

Figure A.1: Probability of Detection in RSNS for $k = 4$

Figure A.2: Probability of Detection in RSNS for $k = 5$

Figure A.3: Probability of Detection in RSNS for $k = 6$

Appendix B

Error Correction Ability of Binary, Gray and Inverse Gray Codes

Figure B.1, B.2 and B.3 compare the error correction capability of binary, Gray and inverse Gray codes under single bit noise perturbation in one, two and three MRSs (without redundancy). The simulation was carried out for several moduli sets of the form $\mathbf{m} = [m-1 \ m \ m+1]$ and several values of bits per residue k (where $k \geq \lceil \log_2(m+2) \rceil$). Subsequently, the complete numerical results are presented in Table A.1.

It can be observed that in general, inverse Gray RSNS (IGRSNS) outperforms binary and Gray RSNS in terms of error correction capability. IGRSNS is able to demonstrate good error correction performance without any redundancy. This is due to the high error detection probability in IGRSNS ($P_D > 0.9$ for any moduli set and number of bits per residue k), which is further supported by the short dynamic range of RSNS as well as the large Hamming distance between neighboring integers in inverse Gray code. To improve the error correction performance further, one redundant modulus was introduced. The proposed error correction algorithm of IGRSNS with one redundant modulus is discussed in Section 4.5.3.

(a) Number of bits per residue $k = 4$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[13 14 15]	75.37	76.26	92.62	42.18	43.29	41.35	22.79	19.44	18.69
[9 10 11]	75.43	76.19	92.35	48.17	50.71	54.95	36.37	35.65	39.84
[7 8 9]	77.76	77.19	96.37	54.4	56.2	71.57	44.03	43.81	61.52
[3 4 5]	74.81	74.61	95.74	57.32	60.03	87.94	57.32	60.03	87.94

(b) Number of bits per residue $k = 5$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[29 30 31]	79.9	82.07	94.1	51.46	49.99	48.7	35.7	31.77	31.77
[23 24 25]	88.74	83.06	97.07	65.23	60.57	64.69	55.23	48.14	55.55
[17 18 19]	88.24	82.9	97.91	67.39	61.22	79.31	62.27	56.38	80.04
[13 14 15]	87.06	82.42	99.74	67.01	62.98	97.25	63.05	60.03	87.14
[9 10 11]	87.04	80.17	100	69.49	63.01	99.4	71.47	64.83	95.33
[7 8 9]	84.66	81.6	100	70.39	67.4	99.64	74.2	70.59	97.52
[3 4 5]	80.78	80.47	99.69	66.6	68.37	99.38	79.07	80.2	99.07

(c) Number of bits per residue $k = 6$

Moduli Set $\mathbf{m}$	Single-bit error, 1 MRS			Single-bit error, 2 MRSs			Single-bit error, 3 MRSs		
	Binary	Gray	IG	Binary	Gray	IG	Binary	Gray	IG
[61 62 63]	81.78	86.1	97.76	68.73	61.23	68.31	61.89	56.97	58.79
[51 52 53]	91.77	87.15	99.37	72.78	65.71	77.33	72.78	65.29	73.24
[41 42 43]	83.4	85.82	97.57	66.49	64.4	82.03	61.01	64.33	74.2
[29 30 31]	90.79	86.55	99.69	72.33	66.77	95.71	68.84	67.68	91.91
[23 24 25]	90.31	86.23	99.87	72.35	70.09	97.07	72.66	72.09	92.59
[17 18 19]	90.12	85.97	99.86	74.72	70.59	98.29	76.92	76.53	95.57
[13 14 15]	89.25	85.51	99.63	74.8	70.67	98.55	79.72	76.66	95.8
[9 10 11]	89.12	85.26	100	75.92	71.51	99.83	83.55	79.43	98.68
[7 8 9]	87.22	84.96	100	75.99	73.42	99.81	86	82.37	99.42
[3 4 5]	84.5	83.72	100	73	73.17	100	88.59	87.32	100

Table B.1: Probability of error correction P_C in RSNS using binary, Gray and inverse Gray (IG) codes

Figure B.1: Probability of Correction in RSNS for $k = 4$ Figure B.2: Probability of Correction in RSNS for $k = 5$

Figure B.3: Probability of Correction in RSNS for $k = 6$

Bibliography

- [1] S. Lin and J. Daniel J. Costello, *Error Control Coding: Fundamentals and Applications*, F. F. Kuo, Ed. Prentice Hall Series in Computer Applications in Electrical Engineering, 1983.
- [2] A. R. Calderbank, “Fifty years of coding theory,” *IEEE Transactions on Information Theory*, vol. 44, no. 6, Oct. 1998.
- [3] A. Omandi and B. Premkumar, *Residue Number Systems: Theory and Implementation*. World Scientific Book, 2007.
- [4] H. Krishna, B. Krishna, K. Y. Lin, and J. D. Sun, *Computational Number Theory and Digital Signal Processing: Fast Algorithms and Error Control Techniques*. CRC Press, 1994.
- [5] R. W. Watson and C. W. Hastings, “Self-checked computation using residue arithmetic,” *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1920–1931, Dec. 1966.
- [6] F. Barsi and P. Maestrini, “Error correcting properties of redundant residue number systems,” *IEEE Transactions on Computers*, vol. 23, no. 3, pp. 307–315, Mar. 1973.
- [7] P. E. Pace, *Advanced techniques for digital receivers*, ser. Artech House radar library, 2000.

- [8] B. L. Luke, "Architecture of an integrated microelectronic warfare system on a chip and design of key components," Ph.D. dissertation, Naval Postgraduate School, Monterey, California, Dec. 2004.
- [9] P. E. Pace, D. Styer, and I. A. Akin, "A folding ADC preprocessing architecture employing a robust symmetrical number system with gray-code properties," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 47, May 2000.
- [10] P. E. Pace, D. Wickersham, D. C. Jenn, and N. S. York, "High-resolution phase sampled interferometry using symmetrical number systems," *IEEE Transactions on Antennas and Propagation*, vol. 49, pp. 1411–1423, Oct. 2001.
- [11] P. E. Pace, D. C. Jenn, and J. P. Powers, "Symmetrical number systems: Theory and applications," *Transworld Research Network Journal*, pp. 91–121, Jan. 2002.
- [12] A. Bouridane, A. Pajayakrit, S. Dlay, and A. Holt, "CMOS VLSI circuits of pipeline sections for 32 and 64-point fermat number transformers," *Integration*, vol. 8, pp. 51–64, 1989.
- [13] S. Gudvangen and A. Patel, "Rapid synthesis of a macro-pipelined CMOS ASIC for the fermat number transform," *Proc. Norwegian Signal Processing Symposium (NORSIG)*, pp. 143–148, Sep. 1995.
- [14] R. V. Kraats and A. Venetsanopoulos, "Hardware for two-dimensional digital filtering using fermat number transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, pp. 155–162, Apr. 1982.

- [15] S. Gudvangen. (1999, Sep.) Practical applications of number theoretic transforms. NORSIQ. [Online]. Available: <http://www.norsig.no/norsig99/Articles/gudvangen.pdf>
- [16] V. Pan, “How can we speed up matrix multiplication?” *SIAM Review*, vol. 26, pp. 393–415, Jul. 1984.
- [17] A. Yagle, “Fast algorithms for matrix multiplication using pseudo-number theoretic transforms,” *IEEE Transactions on Signal Processing*, vol. 43, pp. 71–76, Jan. 1995.
- [18] R. Blahut, “Transform techniques for error control codes,” *IBM J. Res. Develop.*, vol. 23, pp. 299–315, May 1979.
- [19] —, *Theory and practice of error control codes*. Addison-Wesley Publ. Comp., 1984.
- [20] B. Arambepola and S. Choomchuay, “Algorithms and architectures for reed-solomon codes,” *GEC J. of Research*, vol. 9, no. 3, pp. 172–184, 1992.
- [21] L.-L. Yang and L. Hanzo, “A residue number system based parallel communication scheme using orthogonal signaling: Part I - system outline,” *IEEE Transactions on Vehicular Technology*, vol. 51, no. 6, pp. 1534–1546, Nov. 2002.
- [22] —, “A residue number system based parallel communication scheme using orthogonal signaling: Part II - multipath fading channels,” *IEEE Transactions on Vehicular Technology*, vol. 51, no. 6, pp. 1547–1559, Nov. 2002.
- [23] A. S. Madhukumar and F. Chin, “Enhanced architecture for residue number system-based CDMA for high-rate data transmission,” *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, Sep. 2004.

- [24] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. IEEE Press, 1986.
- [25] J. H. McClellan and C. M. Rader, *Number Theory in Digital Signal Processing*. Prentice Hall, 1979.
- [26] W. K. Jenkins and E. J. Altman, "Self-checking properties of residue number error checkers based on mixed radix conversion," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 2, pp. 159–167, Feb. 1988.
- [27] L. L. Yang and L. Hanzo. Coding theory and performance of redundant residue number system codes. [Online]. Available: http://www-mobile.ecs.soton.ac.uk/lly/papers/RRNS_code.pdf
- [28] Y. Wang, "Residue-to-binary converters based on new chinese remainder theorems," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 47, pp. 197–205, Mar. 2000.
- [29] P. E. Pace, R. E. Leino, and D. Styer, "Use of the symmetrical number system in resolving single-frequency undersampling aliases," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1153–1160, May 1997.
- [30] P. E. Pace, J. L. Schaferand, and D. Styer, "Optimum analog preprocessing for folding ADCs," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 42, no. 12, pp. 825–829, Dec. 1995.
- [31] I. Niven, H. S. Zuckerman, and H. L. Montgomery, *An Introduction to the Theory of Numbers*, 5th ed., Jan. 1991.

- [32] P. E. Pace and D. Styer, "High-resolution encoding process for an integrated optical analog-to-digital converter," *Optical Engineering*, vol. 33, pp. 2638–2645, Aug. 1994.
- [33] L.-L. Yang and L. Hanzo, "RNS arithmetic assisted M -ary modulation," *IEEE Communications Letters*, vol. 3, pp. 28–30, Feb. 1999.
- [34] K. Yen, L.-L. Yang, and L. Hanzo, "Residual number system assisted CDMA: A new system concept," *4th ACTS Mobile Communications Summit '99*, pp. 177–182, 1999.
- [35] P. K. Enge and D. V. Sarwate, "Spread spectrum multiple-access performance of orthogonal codes: Linear receivers," *IEEE Transactions on Communications*, vol. COM-35, no. 12, pp. 1309–1319, Dec. 1987.
- [36] A. S. Madhukumar and F. Chin, "Performance of a residue number system based DS-CDMA system over bursty communication channels," *Vehicular Technology Conference (VTC) Fall*, vol. 5, pp. 2433–2440, Sep. 24 – 28, 2000.
- [37] D. Styer and P. E. Pace, "Two channel RSNS dynamic range," *IEEE Transactions on Circuits and Systems—Part I: Fundamental Theory and Applications*, vol. 49, pp. 395–397, Mar. 2002.
- [38] C. J. Kuo, C. H. Lin, and C. H. Yeh, "Noise reduction of VQ encoded images through anti-gray coding," *IEEE Transactions on Image Processing*, vol. 8, no. 1, pp. 33–40, Jan. 1999.
- [39] C. H. Lin and C. J. Kuo, "Channel noise recovery of images through anti-gray coding technique," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 1886–1889, May 1996.

- [40] K. Berberidis, A. Marava, P. Karaivazoglou, and J. Palicot, “Robust and fast converging decision feedback equalizer based on a new adaptive semi-blind channel estimation algorithm,” *Global Telecommunications Conference (GLOBECOM)*, vol. 1, pp. 269–273, Nov. 2001.
- [41] H. Schoeneich and P. A. Hoeher, “Semi-blind pilot-layer aided channel estimation with emphasis on interleave-division multiple access systems,” *Global Telecommunications Conference (GLOBECOM)*, vol. 6, pp. 3513–3517, 2005.
- [42] S. Verdu, “Recent progress in multiuser detection,” in *Advances in Communications and Signal Processing*. Berlin: Springer-Verlag, 1989.
- [43] M. J. Flynn and P. Huang, “Microprocessor design: Thoughts on the road ahead,” *IEEE Micro*, vol. 25, no. 3, pp. 16–31, 2005.