New algorithms and VLSI architectures for efficient RNS computations

Tay, Thian Fatt

2016

Tay, T. F. (2016). New algorithms and VLSI architectures for efficient RNS computations.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/66937>

<https://doi.org/10.32657/10356/66937>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**NEW ALGORITHMS AND VLSI
ARCHITECTURES FOR EFFICIENT RNS
COMPUTATIONS**

**TAY THIAN FATT
SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING**

2016

**NEW ALGORITHMS AND VLSI
ARCHITECTURES FOR EFFICIENT RNS
COMPUTATIONS**

TAY THIAN FATT

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2016

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisor, Associate Professor Chang Chip Hong for his support and guidance during my Ph.D. pursuit in the School of Electrical and Electronic Engineering, Nanyang Technological University. For the past four years, his enthusiasm and passion in research have motivated me to work hard and perform well in research. Without his relentless effort and attitude to strive for the best, I might not have the current research publications and completed this thesis timely. I earnestly appreciate the time spent and the encouragements given by him in the process of writing and reviewing the manuscripts of publications and this thesis.

I would also like to thank Mrs. Chang and Chip's Family members, Li Fei, Hanhua, Jeremy, Zhang Li, Cao Yuan, Zhang Le, Sachin, Siarhei, and Truan. The good times spent with them during gatherings have helped me to release stress and tension from research works.

I am also grateful for the support and assistance from the technical staffs in VIRTUS research centre and VLSI Laboratory. They have been very helpful in providing software support, consultation and a conducive environment for research.

I would like to express my gratitude to Professor Leonel Sousa for hosting and offering me the opportunity to collaborate with him on a mutually beneficial research attachment program in INESC-ID, Lisbon. Without his funding in travel expense and accommodation, I would not have a chance to experience a different research environment. Also, his friendliness and kind support have made my stay in Lisbon an unforgettable wonderful memory.

Table of Contents

Acknowledgment	i
Table of Contents.....	iii
Abstract	vi
List of Figures	vii
List of Tables.....	ix
List of Abbreviations	x
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives.....	5
1.3 Major Contributions.....	6
1.4 Organization of Thesis.....	8
Chapter 2 Background and Literature Review.....	10
2.1 Residue Number System (RNS)	10
2.1.1 Residue Representation.....	10
2.1.2 Residue Arithmetic	12
2.1.3 Residue-to-Binary Conversion	14
2.1.5 Error Detection and Correction in RRNS	17
2.2 RNS Scalers	18
2.3 Error Detection and Correction Algorithms with RRNS	21
Chapter 3 New Algorithm and Architecture for 2^n Scaling of Signed Integer in $\{2^n-1, 2^n, 2^n+1\}$ RNS.....	29
3.1 Introduction	29
3.2 Proposed Scaling Algorithm and Architecture	31
3.2.1 Magnitude Scaling Error in Signed RNS	31
3.2.2 Correction Circuit for Signed RNS Scaling	32

3.2.3	Unified Signed Integer RNS Scaler Architecture	36
3.3	Evaluation and Comparison.....	39
3.3.1	Unit-gate Analysis	39
3.3.2	Logic Synthesis Results	46
3.3.3	Physical Synthesis Results.....	48
3.4	Summary.....	50

Chapter 4 Base Transformation with Injective Residue Mapping for Dynamic Range

Reduction in RNS	52
4.1 Introduction	52
4.2 Injective Mappings for Base Transformation	54
4.3 Proposed Base Transformation Algorithm and Architecture for Dynamic Range Reduction.....	56
4.3.1 Algorithm	56
4.3.2 Architecture	60
4.4 Evaluation and Comparison.....	68
4.4.1 Unit-gate Analysis	69
4.4.2 Synthesis Results	71
4.4.3 Comparison with Removal of One Modulus Method	74
4.5 Summary and Discussions.....	77

Chapter 5 Non-Iterative Single and Multiple Residue Error Detection and Correction

Algorithms.....	79
5.1 Introduction	79
5.2 Single Residue Error Detection and Correction.....	81
5.2.1 Uniqueness of Proposed Error Syndrome	81
5.2.2 Proposed Algorithm and Architecture for Single Residue Digit Error Correction	86
5.2.3 Comparison.....	88
5.3 Multiple Residue Error Detection and Correction	89
5.3.1 Derivation of Syndromes for Error Identification.....	90

5.3.2	Proposed Error Detection and Correction Procedure.....	96
5.3.3	Illustrative Example.....	100
5.3.4	Proof of Uniqueness	102
5.3.5	Error Decoding for Reduced Redundant Space	107
5.3.6	Comparison and Discussions	109
5.4	Summary.....	114
Chapter 6 Conclusion and Future Works		116
6.1	Conclusions	116
6.2	Future Works	117
Author's Publications		120
References		122

Abstract

Residue Number System (RNS) is a non-weighted number system emerging as a promising substitute of two's complement number system for data representation in high-speed and low-power digital signal processing. In RNS, arithmetic operations, such as addition, subtraction and multiplication, can be carried out at a faster speed due to the avoidance of carry propagation. Nonetheless, the arithmetic operations such as sign detection, scaling, base transformation and error decoding, are generally more difficult to be implemented in RNS. This thesis aims to tackle these difficult RNS operations by designing simple and efficient algorithms and architectures. In this thesis, a fast and area efficient 2^n signed integer RNS scaler for the moduli set $\{2^n-1, 2^n, 2^n+1\}$ is proposed. The complex sign detection circuit has been obviated and replaced by simple logic manipulation of some bit-level information of intermediate magnitude scaling results. As a result, the proposed architecture achieves at least 21.6% of area saving, 28.8% of speedup and 32.5% of total power reduction compared to the state-of-art designs. In addition, a new concept of base transformation is introduced to reduce the overall arithmetic processing costs of multi-base RNS. The exploration into this new concept is motivated by the overkill of arithmetic operator sizes in RNS-based digital signal processors (DSPs) due to the choice of moduli set to accommodate for the occasional high precision operations. A new algorithm is proposed to directly map the residues of signed integers from one modulo-arithmetic friendly base $S_1 \equiv \{2^n-1, 2^{n+k}, 2^n+1\}$ to another $S_2 \equiv \{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1\}$, to reduce the word lengths of residue arithmetic operators by α bits. The proposed algorithm utilizes range quantization approach to avoid the complex reverse conversion and sign detection operations. The synthesis results show that its VLSI implementation efficiency, in terms of area-delay product and energy consumption, outperforms the best possible improvised solution. Furthermore, two new algorithms for the correction of single and multiple residue digit errors in Redundant Residue Number System (RRNS) are presented in this thesis. These algorithms involve a fixed number of syndrome computations of to locate the erroneous residue digits. They compare favourably against existing residue digit correction algorithms involving iterative computations as their required computation times are more predictable and thus, they are amenable to hardware implementation.

List of Figures

Figure 2.1 The computations of mixed-radix digit for (x_1, x_2, x_3, x_4)	15
Figure 3.1 Proposed two-stage signed integer RNS scaler.	36
Figure 3.2 Proposed simplified signed integer RNS scaler.	37
Figure 3.3 (a) Modified mod 2^n adder with c_{in} for $n=4$ (b) Simplified mod 2^n adder for $n=4$. ..	38
Figure 3.4 Modified control signal generation module.	39
Figure 3.5 Power-Delay Product for DR = 15, 18, 21 and 24 bits.....	48
Figure 3.6 Layout of the proposed design.	50
Figure 3.7 Layout of the design of [Ye08].	50
Figure 4.1 Base transformation of (a) unsigned integer X ; and (b) signed integer $\tilde{X}$ for $M > M'$	54
Figure 4.2 Mapping of $(x_1, x_2, \dots, x_N)$ to $(x'_1, x'_2, \dots, x'_k)$ for $M > M'$	55
Figure 4.3 Division of M into 2^{n+k} subranges of $2^{2n}-1$ integers.....	57
Figure 4.4 Mapping of valid divisions for base transformation from S_1 to S_2 for $n=5, k=0$ and $\alpha=1$	60
Figure 4.5 Architecture of the proposed base transformation algorithm.....	61
Figure 4.6 Addition stages required for X_{13} computation.....	63
Figure 4.7 Bit slices of MHA1 and MHA2 in Figure 4.6 and MFA in Figure 4.5.....	64
Figure 4.8 Addition stages for the computation of β_L	64
Figure 4.9 Parallel prefix adders with FCI for $n = 4, k = 2$ and $\alpha = 1$	65
Figure 4.10 Architecture in comparison.....	69
Figure 4.11 Percentage reduction of area-delay product.....	72
Figure 4.12 Percentage reduction of power-delay product.....	Error! Bookmark not defined.
Figure 4.13 Architectures of RNS-based inner product step processor (a) dynamic range reduction unexploited (b) dynamic range reduced by the proposed base transformation (c) dynamic range reduced by the removal of one modulus channel.....	76

Figure 4.14 The percentages of reduction on area-delay product by using the proposed design (proposed), RM-1, RM-2, and RM-3 to reduce the dynamic range from $M = 36$ bits to $M' = 27$, 24 and 21 bits with datapath of $N = 10, 15$, and 20 MAC units.....	77
Figure 5.1 Architecture for the proposed residue error correction algorithm.....	88
Figure 5.2 Proposed multiple residue digit error detection and correction procedure.....	100

List of Tables

Table 3.1 Unit-gate Area and Delay of Proposed Design.....	41
Table 3.2 Unit-gate Area of [Ye08]	41
Table 3.3 Estimated Number of Transistors of [Mey03]	43
Table 3.4 Unit-gate Delay of [Mey03]	44
Table 3.5 Number of Transistors and Unit-gate Delay of [Ma10]	45
Table 3.6 Comparison of Number of Transistors and Unit-gate Delay (Value in Parenthesis Indicates the Percentage Reduction of Transistor Count or Delay of Proposed Design Over Its Contender)	45
Table 3.7 Comparison of Synthesized Areas and Delays (Value in Parenthesis Indicates the Percentage Area or Delay Reduction of Proposed Design Over Its Contender).....	47
Table 3.8 Comparison of Leakage and Total Power Dissipations (Value in Parenthesis Indicates the Percentage Leakage or Total Power Reduction of Proposed Design Over Its Contender)...	48
Table 3.9 Comparison of Post-layout Simulation Results.....	49
Table 4.1 Comparison of Unit-gate Area and Delay.	70
Table 4.2 Comparison of Synthesized Area and Delay.	71
Table 4.3 Comparison of Total Power and Leakage Power.	73
Table 5.1 Comparison of Single Residue Error Correction Algorithms.....	89
Table 5.2 Possible Syndrome Expressions for Error Location Categories EL1– EL7	97
Table 5.3 Mapping of Error Location Categories to Syndromes.	97
Table 5.4 Characteristics of Lookup Tables for Error Vector Retrieval.....	99
Table 5.5 Syndrome Values for EL4-EL7	101
Table 5.6 Comparison of Multiple Residue Digit Error Detection and Correction Algorithms.	113
Table 5.7 Comparison of Estimated Unit Delays.....	114

List of Abbreviations

CRT	Chinese Remainder Theorem
CSA	Carry Save Adder
DFT	Discrete Fourier Transform
DR	Dynamic Range
DSP	Digital Signal Processor
DWT	Discrete Wavelet Transform
EAC	End-Around-Carry
FA	Full Adder
FCI	Fast Carry Input
FIR	Finite Impulse Response
FNT	Fermat Number Transform
IDFT	Inverse Discrete Fourier Transform
IEAC	Inverted End-Around-Carry
LSB	Least Significant Bit
LUT	Look-Up Table
MAC	Multiply-Accumulate
MFA	Modified Full Adder
MLD	Maximum Likelihood Decoding
MR	Mixed Radix
MRC	Mixed-Radix Conversion
MSB	Most Significant Bit
RNS	Residue Number System
ROM	Read-Only Memory
RRNS	Redundant Residue Number System
SOC	Systems-On-Chip
SQT	Sum of Quotients Technique
TCS	Two's Complement Number System

Chapter 1

Introduction

1.1 Motivation

There are increasing demands for smaller, faster, and lower power consuming electronic products as in the mobile computing era. Technology scaling has appeared to be a natural approach and arguably one of the most effective means to accommodate these unceasing demands. Currently, device feature size has reached the 14nm technology generation and it is expected to reach the final 5nm physical limit, where the off-leakage current of the entire chip is too large for feasible implementation, after 20-30 years [Iwa09]. The continuous scaling of device is achieved at the cost of the degradation of device reliability. When the feature size becomes smaller, the device is more susceptible to parameter variability. As a result, failure mechanisms subjected to parameter uncertainty will be magnified, which causes the reliability of the device to becoming increasingly unpredictable [Aka11]. The shrinking of device node will also increase the stress factor. Due to the comparatively slower pace of operating voltage scaling, the oxide layer will experience higher electric fields, that will increase its susceptibility to device breakdown [Aka11]. Meantime, the electronic industry is gearing towards the integration of more complicated systems on the same chip without sacrificing their performances. This trend requires more versatile digital signal processor (DSP) to execute complex numerical operations. Current DSPs implemented based on the most commonly used two's complement number system (TCS) face strong hurdles and has reached a bottleneck for further performance improvement despite innovative architectural designs. This is due to the inherent carry propagation chain in the hardware implementation of arithmetic operations in two's complement number system (TCS), that causes the datapaths to be costly, slower, and more power hungry. Therefore, the search for alternative number system is of paramount importance to overcome the limitation.

Residue Number System (RNS) is a non-weighted number system that has received considerable attention lately from researchers. In RNS, an integer with large word-length can be represented using several smaller word-length integers, called residue digits, with respect to the defined moduli set. The arithmetic operations, such as addition, subtraction, and multiplication, can be operated with residue digits independently and in parallel. Since the residue digits are of smaller size, these arithmetic operations suffer from shorter carry propagation delays. Comparing to the positional number systems, such as TCS, the aforementioned arithmetic operations can be carried out at a faster speed in RNS. On the other hand, operations, such as scaling, sign detection, magnitude comparison, and overflow detection, in RNS are more difficult to be implemented efficiently compared to their TCS counterparts due to the presence of inter-modulo computations. Inter-modulo computations require large composite modular operations composing of products of two or more moduli and usually all residue digits across different modulus channels. The loss of independency and parallelism characteristics results in their relatively slower and complex hardware implementations. For this reason, RNS-based arithmetic is more widely applied in the datapaths internally within the application-specific DSPs that are dominated by a large number of additions, subtractions, and multiplications. For example, in [Ram03], a one-dimensional discrete wavelet transform (DWT) architecture based on RNS was proposed. The results of the standard cell based implementation of the RNS-based and TCS-based DWT architectures with 4-tap and 8-tap filter banks show that the RNS-based implementation was approximately 90% faster with slightly larger circuit area. In addition, the RNS-based 16-tap FIR filter proposed in [Con04] was implemented using arithmetic friendly moduli of the forms 2^n and $2^n \pm 1$. The design outperforms its TCS counterpart in area-delay product for the dynamic range spanning from 20 to 40 bits. In [Toi06], the word length restriction of the transform modulus of Fermat number transform (FNT) has been relaxed by decomposing it into two moduli of smaller word lengths. It was found that the arithmetic units can be made more compact and faster than existing solutions of comparable dynamic range.

Despite these conspicuous advantages, the iterative addition and multiplication operations often cause the intermediate computation results to exceed the pre-defined dynamic range and it is imperative to avoid erroneous results due to overflow. The easiest solution is to select the

suitable moduli set upfront to cater for the dynamic range of the largest intermediate operation. However, this method often results in hardware over-utilization due to the exceptionally large-sized operators, which are not required in most other stages of computation. The integration of scalers [Cha11], [Mey03], [Kon09] in the intermediate computation stages is a viable alternative. By scaling the computation results, overflow in subsequent arithmetic operations can be avoided while keeping the hardware simple. More than one scaler are usually required to be effective depending on the number of intermediate computations. Inefficiently implemented scalers will retard the overall speed of the system and introduce remarkable hardware overheads. Therefore, small and fast scaler design is of utmost importance to prevent the scalers from becoming the bottleneck of a system.

Another solution to overflow prevention is to have a multi-base system. Although scaling reduces the dynamic range of the operands, it may introduce precision loss and the arithmetic operation itself does not benefit directly from its reduced operand size. More precisely, the sizes of the residue operators cannot be changed independently to adapt to the scaled operands, unless the moduli that define the residue operators can also be shrunk accordingly. With a single base, there is limited flexibility to trade quality (accuracy) for performance when there is a distinct change in the dynamic ranges of operations. In some applications such as adaptive filtering [Yu02] and fixed coefficient FIR filtering [Fau09], the intermediate results may expand rapidly and then gradually converge to the final output precision. If a smaller moduli set is selected based on the final output precision to conserve area and power consumption of the residue arithmetic operations, more frequent scaling and larger truncation errors may be incurred. With a multi-base RNS, overflow can be minimized to reduce scaling errors in a larger moduli set and the sizes of the modulus operators can be reduced when operating within the smaller moduli set after the intermediate computation results have converged.

Similar to TCS, the positive and negative residue representations in RNS are distributed evenly in the lower and upper half ranges, respectively of the legitimate dynamic range. When dealing with signed residue representation, scaling and base transformation require sign detection upfront as additional correction factor is needed in the computation if the operand in residue

representation is negative. The direct method for detecting the sign of a signed integer in residue representation is to perform a full magnitude recovery by residue-to-binary conversion. This method is inefficient considering that only a single integer border that divides the two ranges is needed to be detected in order to determine the sign. A more effective option is to perform partial magnitude evaluation to detect only the half range resided by the signed residue representation. This range detection method can avoid unnecessary computations in sign detection.

At transistor sizes of a few tens of nano-meters and smaller, aging, noise, radiation, heat and other dynamic conditions can easily perturb a device out of specification and cause transistors operating at low supply voltage to be susceptible to single event upsets and intermittent errors. The dependability of digital systems can be significantly enhanced by incorporating fault tolerance into their design. Unfortunately, classical fault-tolerant techniques such as error correction code, self-checking logic, module replication, reconfiguration, etc. are impractically expensive and non-scalable [Nel90]. Some of these fault diagnosis and reconfiguration approaches to counter soft errors require huge reliable memories or immense test and reconfiguration time. RNS can potentially relax the reliability requirement of low-level circuit by virtue of its parallel and modular arithmetic operations. Due to the isolation of carry forwarding in RNS, variations in circuit power and timing or transient perturbation during computations may alter the states of many devices in proximity but cluster errors occurring in one modulus channel will not be propagated to others [Bar73].

By adding redundant moduli into an existing moduli set, a Redundant RNS (RRNS) is formed, which possesses error detection and correction capabilities by keeping the legitimate range for regular information processing. The residues error is detected by checking if the corresponding magnitude of the received residue representation falls into illegitimate range. Essentially, range detection method is also applicable in residue error detection. Upon confirming their existences, the residue errors have to be first located before correction. Contrary to TCS, error detection and correction methods based on RRNS are able to correct arithmetical processing errors caused by circuit noise, manufacturing defects, process, voltage and temperature variations or

even deliberate fault injection in addition to transmission errors [Tay15b]. Despite its error resilient characteristics, RNS based error detection and correction algorithm generally requires long computation time and large hardware implementation area. This is due to the iterative computations required to locate the erroneous residue digits sequentially. The number of computations is dependent on the size of the moduli set. Since the size of the moduli set escalates with the number of correctable residue errors, a large number of iterative computations is needed for multiple error detection and correction, causing their direct hardware implementation to be as difficult as, if not more complicated than, the implementation of several inter-modulo operations in RNS.

In a nutshell, while RNS has gained a strong foothold and phenomenal successes in domain specific computing, it is not without limitations. The obstacles that constrain its widespread applications are identified to be caused by the different extent of inter-modulo operations. Interesting enough, it is observed that the challenges to overcome the inefficient inter-modulo operations in a number of difficult RNS operations can be strung by a common attribute related to the dynamic range of RNS. This provides us an opportunity to explore theoretic number properties unique to some special moduli and new range quantization approaches to decompose either the size of modular operations of some special moduli sets or reduce the number of iterations and complexity for arbitrary moduli sets to address the deficiencies of existing methods in these computations.

1.2 Research Objectives

Motivated by the possibility of using range detection method to reduce the complexities of arithmetic operations involving simultaneous inter-modulo computations, this research aims to develop efficient algorithms and architectures for signed residue scaling and base transformation. The strategy is to identify a common key denominator to account for the strengths and weaknesses of the existing approaches to reformulate new solutions for special moduli sets whose number theoretic properties can be manipulated for the ease of identification and detection of subranges. RRNS also makes use of range extension to provide error detection

and correction capability for arithmetical processing in residue domain. This is a unique and powerful capability that is missing in other error correction codes used for the reliable delivery or storage of digital data. Limited by the number of balanced and coprime arithmetic friendly moduli, small arbitrary moduli are usually used in RRNS to increase its coding efficiency with minimal impact on area-time complexity to both its regular arithmetic operation on information moduli and error decoding. Considering the time and resource consuming residue error decoding operations with arbitrary moduli sets, this thesis also aims to address the slow RRNS error decoding problem from a different perspective, with special emphasis on preventing the number of sequential steps from increasing linearly with the number of moduli.

To fulfil the above objectives, the following research problems have been identified.

- a) To investigate the potential of using range detection approach to resolve the data dependency of composite moduli so that inter-modulo computations can be carried out in parallel and independently in each modulus channel to a large extent.
- b) To develop an efficient algorithm for scaling signed residue representation without requiring additional sign detection module.
- c) To introduce a new paradigm of multi-base RNS with base transformation for some special moduli sets. As a pioneering effort, a new approach to dynamic range reduction using base transformation will be proposed.
- d) To increase the scalability of RRNS by dismissing the iterative computations for locating residue errors required by all existing residue error detection and correction algorithms.

1.3 Major Contributions

The major contributions of the research work are highlighted as follows:

- a) A new signed integer RNS scaler for moduli set $\{2^n-1, 2^n, 2^n+1\}$ with the scaling factor of 2^n is proposed. Unlike the existing works, the proposed algorithm eliminates the need for a dedicated sign detection module. The sign of the residue representation is obtained merely

by bit manipulations on the intermediate computation results of the scaler. The proposed architecture is designed using adders and simple logic gates without any read-only memory (ROM). Due to its simplicity, the proposed architecture is able to achieve a faster speed, smaller hardware area, and lower power consumption than the state-of-art designs with the same scaling factor and comparable dynamic range.

- b) A pioneering concept of using base transformation to reduce the sizes of residue arithmetic operations along with dynamic range reduction in a multi-base RNS is introduced. In this new design paradigm, dynamic range is contracted by trimming all the moduli simultaneously, which results in a greater hardware cost savings while preserving the parallelism and modularity of the original base. A new algorithm for remapping the residue representation of a signed integer from the moduli set $S_1 = \{2^n - 1, 2^{n+k}, 2^n + 1\}$ to another three-moduli set $S_2 = \{2^{n-\alpha} - 1, 2^{n+k-\alpha}, 2^{n-\alpha} + 1\}$ is proposed, where k and α are integers in the ranges, $[0, n]$ and $[1, n-2]$, respectively. Since the parameters, n , k , and α can be flexibly chosen, there is a rich combination of S_1 and S_2 to fit into a wide-range of applications with different dynamic range requirements. The proposed algorithm employs range quantization method to avoid unnecessary computations when remapping residue representation and simplifies sign detection. Compared to the best possible improvised solution, the proposed architecture manages to achieve significant savings, especially when there is a large reduction in dynamic range.
- c) A new algorithm for single residue error detection and correction in RRNS using two redundant moduli is proposed. The erroneous residue digit can be identified by retrieving the location and magnitude of the error digit from a lookup table using the syndrome computed from the received residue representation. The syndrome is proven to be unique for every possible residue error with two simple requirements imposed on the selection of redundant moduli. Another new algorithm for detecting and correcting multiple residue errors is proposed. By dividing the received residue digits into three groups, any residue errors can be identified by one of the seven error location categories. Three syndromes are computed using the three groups of residue digits, which represent the extracted magnitude

of the residue digit errors. The exact locations and magnitudes of the error digits are retrieved from six lookup tables consuming no more than three cycles using the syndromes as inputs. In contrast to the existing algorithms, the number of computations for locating error is fixed regardless of the size of the chosen moduli set. The uniqueness of the syndromes can be assured by imposing two simple criteria. One of the criterion can be further relaxed to improve the code rate by using more than three syndromes to identify the residue errors.

The above contributions have led to the publications listed in the author's publications at the end of the thesis.

1.4 Organization of Thesis

This thesis is organized into six chapters. Chapter 1 presents the motivations, objectives, and major contributions achieved in this research.

In Chapter 2, the fundamentals of RNS including residue representation, residue arithmetic operations, residue-to-binary conversion, and residue error detection and correction in RRNS are described in detail. In addition, extensive reviews of the works on RNS scaler and residue error detection and correction algorithms are also presented.

The main contributions of this thesis are detailed in Chapters 3 to 5. Chapter 3 presents a new 2^n signed integer scaler for moduli set $\{2^n-1, 2^n, 2^n+1\}$. The detection and correction of magnitude scaling error in signed RNS is first discussed. Subsequently, the proposed algorithm and simple adder-based architecture are presented. The proposed design and the state-of-art designs are analyzed and compared based on unit-gate model, logic synthesis and physical implementation results.

In Chapter 4, the novel concept of injective residue mapping for base transformation is introduced. Based on this concept, a new algorithm for remapping the residue representation of

a signed integer from the moduli set $\{2^n-1, 2^{n+k}, 2^n+1\}$ to $\{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1\}$ is proposed. The proposed architecture is implemented by applying the arithmetic properties of $2^n \pm 1$ discussed in Chapter 2. As this is the first ever residue representation remapping module reported in literature, the proposed design is evaluated against the best improvised solution comprised of residue-to-binary converter, comparator with constant, and residue generators. The comparison is carried out using unit-gate analysis and logic synthesis with several chosen values of n , k , and α .

A new single residue error detection and correction algorithm is proposed in Chapter 5. The uniqueness proof of the proposed algorithm is first shown before the presentation of the proposed procedure and architecture. The proposed algorithm is compared with existing works in terms of the need for iterative computation, output domain, and moduli selection criteria. In addition, a new multiple residue error detection and correction algorithm is also presented. The three syndromes are derived based on the categorization of received residue digits into three distinct groups. The feasibility of the error detection and correction procedure is verified by using an illustrative example and the syndrome uniqueness proof. To reduce the redundant moduli space, an extended version of the proposed algorithm with variable number of syndromes is also proposed. Both algorithms are compared against existing algorithms based on the number of correctable errors, methods for detecting and locating residue errors, and the error correction output domain.

Chapter 6 summarizes the key contributions discussed in Chapters 3 to 5, and presents the potential research topics for future exploration.

Chapter 2

Background and Literature Review

2.1 Residue Number System (RNS)

2.1.1 Residue Representation

Numbers were created by ancient human to count and measure objects. A numeral system is a way to represent the numbers uniquely where algebraic and arithmetic operations acted upon them can be carried out without ambiguity. Generally, there are two types of numeral systems, i.e., weighted and non-weighted number systems. In a weighted number system with a set of weights, w_i , any number X can be expressed by

$$X = \sum_{i=1}^n a_i w_i \quad (2.1)$$

where a_i is a digit at the i -th position.

The most commonly used weighted number system in daily life is decimal number system. The w_i 's of decimal number system are the successive powers of radix 10 and a_i is a digit in the symbol set $\{0, 1, \dots, 9\}$. Meanwhile, binary number system is a weighted number system widely used in computer arithmetic. It has a radix of 2 and a_i is a digit in the symbol set $\{0, 1\}$. Modern-day digital processors are built using transistors which operate on the principle of two possible states, 'off' and 'on'. Since binary number system has only two possible digits, '0' and '1', it is suitable to be implemented in digital signal processors where '0' and '1' represent 'off' and 'on' states, respectively.

On the other hand, RNS is a non-weighted number system. An RNS is characterized by its base, which is a set of pair-wise relatively prime integers $\{m_1, m_2, \dots, m_k\}$, where m_i is called modulus.

An unsigned integer X within the range of $[0, M_K - 1]$ can be uniquely represented by a k -

tuple $(x_1, x_2, \dots, x_k)$, where the dynamic range, $M_K = \prod_{i=1}^k m_i$. The residue x_i is the least positive remainder of the division of X by m_i , and is usually represented as $X \bmod m_i$ or $|X|_{m_i}$.

In RNS $\{m_1, m_2, \dots, m_k\}$, the residue representation of an integer $X' \in [C, M_K + C - 1]$ for any integer constant C can be mapped to the residue representation of an integer $X \in [0, M_K - 1]$ by using the following congruence [Tom08]:

$$X' \equiv |X - C|_{M_K} + C \quad (2.2)$$

For the sign representation in RNS, C is chosen to be any integer in the range $(-M_K, 0)$. Equation (2.2) makes it possible to represent signed numbers in residue form while keeping the arithmetic units for the operations $\{+, -, \times\}$ compatible with those of the unsigned residues under the same RNS. If C is set to $-\lfloor \frac{M_K}{2} \rfloor$ where $\lfloor y \rfloor$ denotes the largest integer less than or equal to y , an almost symmetrical (for even M_K) or a symmetrical (for odd M_K) range of signed integers can be obtained. Under this signed residue representation, a signed integer $\tilde{X}$ in the range $\left[-\frac{M_K}{2}, \frac{M_K}{2} - 1\right]$ if M_K is even or $\left[-\frac{M_K - 1}{2}, \frac{M_K - 1}{2}\right]$ if M_K is odd [Su90] can be uniquely represented by a k -tuple $(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$. Substituting $X' = \tilde{X}$ and $C = -\lfloor M_K/2 \rfloor$, Equation (2.2) can be rewritten as

$$\tilde{X} \equiv \left| X + \left\lfloor \frac{M_K}{2} \right\rfloor \right|_{M_K} - \left\lfloor \frac{M_K}{2} \right\rfloor \quad (2.3)$$

When $\tilde{X} \geq 0$, the residue representation of X can be mapped to that of $\tilde{X}$ in the range of $\left[0, \frac{M_K}{2} - 1\right]$, if M_K is even and $\left[0, \frac{M_K - 1}{2}\right]$, if M_K is odd; When $\tilde{X} < 0$, the residue representation of X can be mapped to that of $\tilde{X}$ in the range of $\left[\frac{M_K}{2}, M_K - 1\right]$ if M_K is even

and $\left[\frac{M_K+1}{2}, M_K-1\right]$ if M_K is odd. Using this mapping, the relationship between $\tilde{X}$ and X

under the same moduli set can be written as:

$$\begin{aligned}\tilde{X} &= \begin{cases} X & \text{if } \tilde{X} \geq 0 \\ X - M_K & \text{if } \tilde{X} < 0 \end{cases} \\ \tilde{X} &= X - s \cdot M_K\end{aligned}\tag{2.4}$$

where $s \in \{0, 1\}$ represents the sign of $\tilde{X}$.

The sign s of $\tilde{X}$ can be determined by checking if X falls in the lower or upper half range of M_K [Etz80], [Tay13]. When M_K is even,

$$s = \begin{cases} 0 & \text{if } X \in [0, (M_K/2)-1] \\ 1 & \text{if } X \in [M_K/2, M_K-1] \end{cases}\tag{2.5}$$

On the other hand, when M_K is odd,

$$s = \begin{cases} 0 & \text{if } X \in [0, (M_K-1)/2] \\ 1 & \text{if } X \in [(M_K+1)/2, M_K-1] \end{cases}\tag{2.6}$$

2.1.2 Residue Arithmetic

In an RNS with moduli set $\{m_1, m_2, \dots, m_k\}$, the integers, X and Y can be represented by

$(x_1, x_2, \dots, x_k)$ and $(y_1, y_2, \dots, y_k)$, respectively. Let $Z = X \circ Y$ represents the result of the

arithmetic operations acted upon X and Y . Z can be expressed as follows:

$$\begin{aligned}X \circ Y &\equiv (|x_1 \circ y_1|_{m_1}, |x_2 \circ y_2|_{m_2}, \dots, |x_k \circ y_k|_{m_k}) \\ Z &\equiv (z_1, z_2, \dots, z_k)\end{aligned}\tag{2.7}$$

where $z_i = |x_i \circ y_i|_{m_i}$ and ‘ $\circ$ ’ represents an arithmetic operation, such as addition, subtraction, multiplication, squaring, or exponentiation.

From (2.7), the residue digits of Z , $(z_1, z_2, \dots, z_k)$, are computed by performing arithmetic operation on the residue digits of X and Y in the respective modulus channels. Therefore, the residue arithmetic operation in each modulus channel can be performed in parallel and independently without carry-propagation across modulus channels. Since the word-lengths of the residue digits are considerably shorter than those of the corresponding integers, the carry

propagation delays of the residue arithmetic operations are lower than those of arithmetic operations performed in weighted number systems. Nonetheless, the operations such as scaling, sign detection, magnitude comparison, and base extension are more difficult to be implemented in RNS because these operations involve inter-modulo computations. An inter-modular operation is one that requires more than a residue and product of several moduli to compute. As inter-modulo computations are time-consuming, the aforementioned operations are more difficult to be implemented efficiently compared to their weighted number system counterparts.

Residue arithmetic, however, is richer in theoretic number properties. The following key identities and properties of residue arithmetic are used extensively in this thesis:

Property 2.1: $|xm|_m = 0$

Property 2.2: $|ax|_m = a|x|_m$

Property 2.3: $|-x|_m = |m - x|_m$

Property 2.4: $x - |x|_m = \left\lfloor \frac{x}{m} \right\rfloor \cdot m$

Property 2.5: The mod $2^n - 1$ multiplication of an integer by a powers-of-two integer can be obtained by a circular left shift operation [Cha11]:

$$|2^r y|_{2^n - 1} = CLS_n(y, r) \quad (2.8)$$

where $CLS_n(y, r)$ represents the circular left shift operation of an n -bit binary representation of an integer y by r bit positions, and $0 \leq r < n$.

Property 2.6:

$$|-2^r y|_{2^n - 1} = |2^r (2^n - 1 - y)|_{2^n - 1} = |2^r \bar{y}|_{2^n - 1} = CLS_n(\bar{y}, r) \quad (2.9)$$

where $\bar{y}$ represents the one's complement of y .

Property 2.7: The mod $2^n + 1$ multiplication of an n -bit integer y by two to the power of r ($0 \leq r < n$) is equivalent to the mod $2^n + 1$ addition of the one's complement of the left shifted y by r positions with a correction factor of $-(2^r - 1)$ [Mur12].

$$\begin{aligned}
|2^r y|_{2^n+1} &= |2^r (2^{n-1} y_{n-1} + 2^{n-2} y_{n-2} + \dots + 2^0 y_0)|_{2^n+1} \\
&= \left| 2^{n+r-1} y_{n-1} + 2^{n+r-2} y_{n-2} + \dots + 2^r y_0 \right|_{2^n+1} \\
&= |2^{r-1} \bar{y}_{n-1} + 2^{r-2} \bar{y}_{n-2} + \dots + 2^r \bar{y}_0 - (2^{r-1} + 2^{r-2} + \dots + 1)|_{2^n+1} \\
&= |CCLS_n(y, r) - (2^r - 1)|_{2^n+1}
\end{aligned} \tag{2.10}$$

where y_i denotes the i -th bit of the binary representation of y and $CCLS_n(y, r)$ denotes the complementary circular left shift operation of the n -bit word y by r bits.

Property 2.8: The mod 2^n+1 negation of y can be expressed as [Tso13]:

$$|-y|_{2^n+1} = |2^n - 1 - y + 2|_{2^n+1} = |\bar{y} + 2|_{2^n+1} \tag{2.11}$$

2.1.3 Residue-to-Binary Conversion

Since RNS is a non-weighted number system, the position of a residue digit does not represent its weight and the magnitude of a residue representation cannot be computed as straightforwardly as (2.1). The magnitude of an integer can be evaluated from its residue representation by using Mixed-Radix Conversion (MRC) or Chinese Remainder Theorem (CRT). Given the residue representation $(x_1, x_2, \dots, x_k)$, the integer X can be computed using the MRC as follows:

$$X = \sum_{i=1}^k a_i \prod_{j=1}^{i-1} m_j + a_1 \tag{2.12}$$

where a_i is called mixed-radix digit and it can be computed from $(x_1, x_2, \dots, x_k)$ by

$$\begin{aligned}
a_1 &= x_1 \\
a_2 &= \left| (x_2 - a_1) \right|_{m_1}^{-1} \Big|_{m_2} \\
a_3 &= \left| \left((x_3 - a_1) \right|_{m_1}^{-1} - a_2 \right) \right|_{m_2}^{-1} \Big|_{m_3} \\
&\dots \\
a_k &= \left| \left(\left((x_k - a_1) \right|_{m_1}^{-1} - a_2 \right) \right|_{m_2}^{-1} - \dots - a_{k-1} \right) \right|_{m_{k-1}}^{-1} \Big|_{m_k}
\end{aligned} \tag{2.13}$$

The MRC involves sequential computations whereby the evaluation of a_i , is dependent on a_{i-1} as shown in Equation (2.13). Figure 2.1 shows the computation of the mixed-radix digits of (x_1, x_2, x_3, x_4) . The computations of a_i for $0 < i \leq 4$ can only be carried out before a_{i-1} is evaluated.

Therefore, the computation of a_4 has the longest delay among those of other mixed-radix digits. Due to the sequential computations of mixed-radix digits, the MRC is not suitable for magnitude evaluation when the selected moduli set is large as the computation time increases linearly with the number of moduli.

Figure 2.1 The computations of mixed-radix digit for (x_1, x_2, x_3, x_4)

On the other hand, X can also be evaluated by the CRT from its residue representation $(x_1, x_2, \dots, x_k)$ as follows:

$$X = \left| \sum_{i=1}^k M_i \left| M_i^{-1} \right|_{m_i} x_i \right|_{M_K} = \sum_{i=1}^k M_i \left| \left| M_i^{-1} \right|_{m_i} x_i \right|_{m_i} - r_k M_K \quad (2.14)$$

where $M_i = M_K / m_i$, $\left| M_i^{-1} \right|_{m_i}$ is the multiplicative inverse of $\left| M_i \right|_{m_i}$, and r_k is an integer in the range $[0, k-1]$. The following example shows the computation of X from its residues by using CRT.

Example 2.1: Consider the moduli set $\{m_1, m_2, m_3\} \equiv \{7, 8, 9\}$. The dynamic range of the moduli set can be calculated as $M_K = 7 \times 8 \times 9 = 504$. Since $M_1 = 504/7 = 72$, $M_2 = 504/8 = 63$, and $M_3 = 504/9 = 56$, the corresponding multiplicate inverses can be computed as $\left| M_1^{-1} \right|_{m_1} = 4$, $\left| M_2^{-1} \right|_{m_2} = 7$, and $\left| M_3^{-1} \right|_{m_3} = 5$, respectively. Given a residue representation $X \equiv (x_1, x_2, x_3) \equiv (0, 5,$

2). X can be computed by using CRT as follows:

$$\begin{aligned} X &= \left| M_1 \left| M_1^{-1} \right|_{m_1} x_1 + M_2 \left| M_2^{-1} \right|_{m_2} x_2 + M_3 \left| M_3^{-1} \right|_{m_3} x_3 \right|_{M_K} \\ &= \left| (72 \times 4 \times 0) + (63 \times 7 \times 5) + (56 \times 5 \times 2) \right|_{504} \\ &= 245 \end{aligned} \quad (2.15)$$

In the CRT, the computation of each partial sum, $M_i \left\| M_i^{-1} \right\|_{m_i} x_i \Big|_{m_i}$ can be performed in parallel and at a faster speed compared to the sequential computations of mixed-radix digits in the MRC. Nonetheless, after summation of the partial sum, $M_i \left\| M_i^{-1} \right\|_{m_i} x_i \Big|_{m_i}$, the result has to be modularly reduced by M_K to ensure that the final result X is in the range $[0, M_K-1]$. Since M_K usually is a large integer, the modulo reduction operation is difficult to be implemented efficiently on hardware.

To overcome the large modulo reduction problem in the CRT and slow sequential computation problem in the MRC, respectively, two improved conversion algorithms, namely New CRT-I and New CRT-II were proposed in [Wan00]. The New CRT-I uses the weighted coefficient concept in the MRC and introduces a different mixed-radix representation y_i , which can be computed in parallel from the residue digits:

$$\begin{aligned} X &= \left| x_1 + p_1 m_1 (x_2 - x_1) + p_2 m_1 m_2 (x_3 - x_2) + \dots + p_{(k-1)} m_1 m_2 \dots m_{k-1} (x_k - x_{k-1}) \right|_{m_1 m_2 \dots m_k} \\ X &= \left| y_0 + y_1 m_1 + y_2 m_1 m_2 + \dots + y_{(k-1)} m_1 m_2 \dots m_{k-1} \right|_{m_1 m_2 \dots m_k} \end{aligned} \quad (2.15)$$

where

$$\begin{aligned} \left| p_1 m_1 \right|_{m_2 m_3 \dots m_k} &= 1 \\ \left| p_2 m_1 m_2 \right|_{m_3 m_4 \dots m_k} &= 1 \\ &\vdots \\ \left| p_{(k-1)} m_1 m_2 \dots m_{k-1} \right|_{m_k} &= 1 \end{aligned} \quad (2.16)$$

The approach taken by the New CRT-I solves the slow computation problem of mixed-radix digit in the MRC. However, the calculation of X still involves large modulo reduction operation which is costly in hardware implementation. On the other hand, the New CRT-II uses parallel prefix tree concept whereby the modulo reduction operation in each prefix computation is of smaller size. The magnitude of residue representation $(x_1, x_2, \dots, x_k)$ can be evaluated using the following algorithm [Wan00]:

Algorithm: $\text{translate} \left((x_1, x_2, \dots, x_k), X \right)$

1) if $n = 2t > 2$ (n is an even number greater than 2), then

$$\text{translate} \left((x_1, x_2, \dots, x_t), N_1 \right), P_1 = \prod_{i=1}^t m_i$$

$$\text{translate} \left((x_{t+1}, \dots, x_k), N_2 \right), P_2 = \prod_{i=t+1}^k m_i$$

$$\text{findno} (N_1, N_2, P_1, P_2, X).$$

2) if $n = 2t + 1 > 2$ (n is an odd number greater than 2), then

$$\text{translate} \left((x_1, x_2, \dots, x_t), N_1 \right), P_1 = \prod_{i=1}^t m_i$$

$$\text{translate} \left((x_{t+1}, \dots, x_k), N_2 \right), P_2 = \prod_{i=t+1}^k m_i$$

$$\text{findno} (N_1, N_2, P_1, P_2, X).$$

3) if $n = 2$, then (x_1, x_2, m_1, m_2, X) .

4) if $n = 1$, then $X = x_1$

Procedure: $\text{findno} (x_1, x_2, m_1, m_2, X) :$

1) find a p_0 such that $|p_0 m_2|_{m_1} = 1$.

2) $X = x_2 + |p_0 (x_1 - x_2)|_{m_1} m_2$.

The magnitude can be evaluated in t (n is even) or $t + 1$ (n is odd) stages of prefix computation.

This method involves smaller modulo reduction operation in each prefix computation than the CRT and requires less stages of computation than the MRC.

2.1.5 Error Detection and Correction in RRNS

By expanding the code space of a residue representation to include r additional residue digits, an (n, k) maximum distance separable code can be formed using an RRNS. An RRNS has a base of $n = k + r$ coprime moduli $\{m_1, m_2, \dots, m_k, m_{k+1}, \dots, m_n\}$ and a dynamic range of

$M_N = \sum_{j=1}^n m_j$. The additional moduli $m_{k+1}, m_{k+2}, \dots, m_{k+r}$ are called the redundant moduli as opposed to the information moduli $m_1, m_2, \dots, m_k$ of the original base. The interval $[0, M_k-1]$ of the k information residue digits of the n -tuple is called the legitimate range and the interval $[M_k, M_N-1]$ due to the remaining r redundant residue digits is called the illegitimate range. The legitimate range represents the useful computational range of the number system [Ort92] whereas the illegitimate range is useful for error and overflow detection [Etz80].

An integer X within the legitimate range is converted to $X' = X + E$ when in error, where E is the error magnitude. X' always falls in the illegitimate range if $m_{k+1}, m_{k+2}, \dots, m_{k+r} > m_i \forall i \leq k$ [Goh08]. Thus, from the magnitude of X' , the errors on the residues can be detected and then corrected. An RRNS with r redundant moduli is capable of detecting r and correcting $\lfloor r/2 \rfloor$ residue digit errors [Goh08].

2.2 RNS Scalers

The first RNS scaling algorithm was proposed in [Sza67] and the scaling constant is the product of a subset of moduli of the moduli set. This algorithm is based on computational intensive base extension method and is implemented using look-up tables. The total processing time is k clock cycles, where k is the number of moduli. In subsequent years, many new techniques have been proposed and they can be categorized into scaling schemes with fixed scaling constant [Gri88], [Gri89], [She89a], [Bar95], [Ulm98], [Gar99], [Ye08], [Kon09], [Ma10], [Cha11] and with programmable powers-of-two scaling constant [Mey03], [Car05], [Low12].

Two CRT based scaling algorithms namely, L -CRT and 2^{2n-q} -CRT were proposed in [Gri88]. The L -CRT scheme was designed for general moduli set with scaling error bounded by L , where L is defined as the number of moduli in [Gri88]. The scaling constant of the scheme V has to be chosen such that $M = VM'$, where M is the dynamic range and $M' = 2^i < M$ for $i > 0$. The rationale of choosing such scaling constant is to enable the CRT computation shown in Equation (2.14) to be performed using binary adder instead of using modulo M adder. The 2^{2n-q} -

CRT was designed for the special moduli set $\{2^n-1, 2^n, 2^n+1\}$ whereby the dynamic range M is approximated by 2^{3n} or $3n$ bits. Similar to the consideration in L -CRT scheme, the scaling constant of this scheme has to be an integer with magnitude 2^q for $q < 2n$. Later, the same authors proposed another scheme in [Gri89] with a less stringent constraint on the choice of scaling constant. The scaling constant can be any positive number less than M . However, the relaxation of the constraint is achieved at the cost of potentially larger error.

The scheme proposed in [She89a] was formulated based on the CRT with a scaling error of at most unity and requires a redundant moduli for the computation of r_k in (2.14). This scheme is implemented using look-up tables requiring approximately $\log_2 n$ clock cycles, where n is the total number of moduli in the RNS. Since it involves redundant moduli, the arithmetic units prior to the scaler require additional hardware area for the computation of the redundant residue digit.

According to [Bar95], the scaled result of an integer, X by a factor F_s can be expressed as:

$$X_s = \left\lfloor \frac{X - |X|_{F_s}}{F_s} \right\rfloor \frac{M}{F_s} \quad (2.17)$$

where $F_s = \prod_{i=1}^s m_i$, $M = \prod_{i=1}^n m_i$ and $s < n$. The residue representation of X_s can be computed

by performing base extensions on $|X|_{F_s}$ and $\left\lfloor \frac{X - |X|_{F_s}}{F_s} \right\rfloor \frac{M}{F_s}$. Base extension is essentially a

transformation of the residues in one RNS base to equivalent residues in an expanded RNS base. The base extension method used in this scheme is formulated based on the CRT and does not require any redundant modulus. This is achieved at the expense of a more complicated hardware implementation.

The scaling algorithm proposed in [Ulm98] is another CRT based scheme. The scaled integer is computed by summing the truncated partial product in Equation (2.14). Thus, a scaling error bounded by the number of moduli is introduced into the scaling result but the error can be reduced to 1 or 1.5 if a correction circuit is employed. This scheme is implemented using a

mixture of ROM and multioperand adders. Due to the ROM based design, this scheme is not suitable for high-speed pipelined applications.

Unlike the previous schemes involving iterative computations, the scaling scheme proposed in [Gar99] requires only two look-up cycles. Nonetheless, this scheme is only suitable for the RNS with a small number of moduli because the number of inputs of each look-up table is dependent on the number of moduli. For the RNS with large number of moduli, the look-up tables required in this scheme are not viable for hardware implementation.

The scheme proposed in [Ye08] was designed for scaling signed integer in the special moduli set $\{2^n-1, 2^n, 2^n+1\}$ with 2^n as the scaling constant. The architecture of the proposed scheme consists of an unsigned integer scaler, a correction circuit and a dedicated RNS sign detection circuit. According to the derivations using Equation (2.17), the scaling of a signed integer $\tilde{X}$ involves the addition of a correction factor of '1' to the scaled residue digit in 2^n modulus channel if $\tilde{X} < 0$. The same authors proposed another signed integer scaling scheme for general moduli set with the scaling constant of 2^n [Ma10]. This scheme uses base extension method to compute the scaled residue digits before adding the necessary correction factors if the signed integer is less than zero. Similar to that in [Ye08], this scheme also requires a dedicated RNS sign detection circuit. Since the dedicated RNS sign detection circuit requires large hardware area, these two schemes are not suitable for applications that have a tight constraint on hardware area.

In [Kon09], the scaling constant can be any number coprime to the moduli and the proposed scheme uses similar approach as [She89a], [Bar95] and [Gar99]. However, this scheme requires base extensions on only $|X|_{F_i}$ in Equation (2.17) because the scaling constant is not the product of the subset of moduli in the RNS. Therefore, it requires less hardware area and has almost the same time complexity as those of [She89a], [Bar95] and [Gar99].

The CRT based scaling algorithm proposed in [Cha11] is also designed for the special moduli set $\{2^n-1, 2^n, 2^n+1\}$ with a scaling constant of 2^n . By exploiting theoretic number properties of

RNS, the residue digits of the scaled integer can be computed by using adders only and thus, its hardware implementation is faster and smaller than the other schemes. Unlike the other schemes formulated based on the CRT, there is no scaling error introduced into the scaled integer using this scheme.

The scaling schemes proposed in [Mey03], [Gri89] and [She89a] are designed for programmable powers-of-two scaling constant. They are formulated based on either division remainder zero [Mey03], [Car05] or the CRT [Low12]. For the schemes formulated based on division remainder zero, the moduli have to be chosen to be odd and pairwise relatively prime integers. According to [Mey03], the residue digits of an integer X being scaled have to be incremented by 1 if X is odd before multiplied by the multiplicative inverse of two with respect to the respective modulus. The programmable powers-of-two scaler is built by cascading scaling-by-two blocks whereby each block consists of modulo incrementer circuits, multiplexers, multipliers, a sign detection circuit and a parity detection circuit. The scheme proposed in [Car05] is essentially the same as that of [Mey03]. The authors of [Car05] extended the works of [Mey03] by describing the implementations of the sign detection and parity detection circuits in detail. As suggested in [Car05], the sign detection and parity detection circuits can be implemented by using the methods suggested in [Mil85] and [Lu92], respectively with an additional redundant modulus. On the other hand, the scheme in [Low12] exploits the numeral properties of RNS and the CRT to formulate an efficient programmable powers-of-two scaling algorithm for the moduli set $\{2^n-1, 2^n, 2^n+1\}$. The computations in each modulus channel can be carried out in parallel by using at most two logarithmic shifters, one-stage of carry save adder and a modulo adder. Due to its simple architecture, it is faster and smaller than the only available and most efficient hybrid programmable power-of-two scaler for the same moduli set.

2.3 Error Detection and Correction Algorithms with RRNS

The concept of error detection and correction in RRNS was first introduced in [Wat66]. By adding two redundant moduli, m_{k+1} and m_{k+2} into the RNS with bases $\{m_1, m_2, \dots, m_k\}$, the

algorithm proposed in [Wat66] is able to detect and correct single residue digit error with the following conditions: (1) m_{k+1} is greater than any information moduli; (2) m_{k+2} is more than twice as large as any of the information moduli; and (6.18) $\prod_{i=k+1}^{k+2} m_i > 2 \sum_{i=1}^k (m_i - 1) + \sum_{i=k+1}^{k+2} (m_i - 1)$.

The proposed algorithm introduces a 'consistency checking' method to detect erroneous residue digit. Firstly, base extension method is used to compute x_{k+1} and x_{k+2} with respect to the received information residue digits. Then, x_{k+1} and x_{k+2} are compared with the received redundant residue digits and the differences are computed as $|\Delta|_{m_{k+1}}$ and $|\Delta|_{m_{k+2}}$, respectively.

If both Δ_1 and Δ_2 are zeros, there is no erroneous residue digit. Otherwise, if either $|\Delta|_{m_{k+1}}$ or $|\Delta|_{m_{k+2}}$ is non-zero, it can be concluded that the received residue digit of the corresponding non-zero $|\Delta|_{m_i}$ is erroneous and it is replaced with the computed residue digit. If both $|\Delta|_{m_{k+1}}$ and $|\Delta|_{m_{k+2}}$ are non-zero, one of the information residue digits is erroneous. The position of the erroneous residue digit and error digit can be found by using $|\Delta|_{m_{k+1}}$ and $|\Delta|_{m_{k+2}}$ as an address to a look-up table. The erroneous residue digit can be corrected by modularly subtracting the error digit.

The authors in [Sza67] proved that whenever there is an erroneous residue digit in an RRNS, the magnitude of the received residue representation will always be outside the range

$$\left[-\frac{M_K}{2}, \frac{M_K}{2} - 1 \right], \text{ where } M_K \text{ is the product of the information moduli. Let}$$

$X \equiv (x_1, \dots, x_p, \dots, x_k, x_{k+1})$ be the error-free residue representation and

$X' \equiv (x_1, \dots, x'_p, \dots, x_k, x_{k+1})$ be the residue representation with an erroneous residue digit

located in some modulus channel m_p . The difference between X and X' can be computed as

$$a \left(\frac{M_K \cdot m_{k+1}}{m_p} \right), \text{ where } a \text{ is a positive integer less than } m_p. \text{ Since } a \left(\frac{M_K \cdot m_{k+1}}{m_p} \right) \geq M_K \text{ for}$$

$m_{k+1} \geq m_p$, X' always falls outside the range $\left[-\frac{M_K}{2}, \frac{M_K}{2} - 1 \right]$ if X is within the range

$\left[-\frac{M_K}{2}, \frac{M_K}{2}-1\right]$. A single residue error correction algorithm was proposed in [Sza67] by using

two redundant moduli. The mixed-radix equivalent of X' is first computed. If X' falls in the

range $\left[-\frac{M_K}{2}, \frac{M_K}{2}-1\right]$, there is no detectable error. Otherwise, there exists an erroneous residue

digit, and all the numbers that are at a distance of $a\left(\frac{M_K \cdot m_{k+1} \cdot m_{k+2}}{m_i}\right)$ from X' are determined.

X can then be determined by finding the values of i and a such that $\left|X' - a\left(\frac{M_K \cdot m_{k+1} \cdot m_{k+2}}{m_i}\right)\right|_{M_K}$

falls in the range $\left[-\frac{M_K}{2}, \frac{M_K}{2}-1\right]$.

The author of [Man72] extended the work of [Sza67] to prove that when there are $t \leq \lfloor r/2 \rfloor$ residue digit errors, the magnitude of the residue representation always falls in the range $[M_K, M_N - 1]$ where r is the number of redundant moduli. In [Man72], a single residue error detection and correction method was proposed. The range of error magnitude is first estimated by computing the difference between the magnitude of the erroneous residue representation and M_K . Then, the erroneous residue digit can be located by referencing the computed range of error magnitude to a pre-computed error detection table.

In [Yau73], two residue error correction algorithms were presented, one for the single and one for the multiple residue digit error correction. The proposed single residue error correction algorithm uses a similar method to that presented in [Wat66], which relies on the base extension for consistency check. Instead of using lookup table for locating the erroneous residue digit, the proposed algorithm requires the computation of the differences between the received and the base extended information residue digits, $|\Delta|_{m_i}$ for $i=1, 2, \dots, k$ in addition to $|\Delta|_{m_{k+1}}$ and $|\Delta|_{m_{k+2}}$ computations in [Wat66]. The proposed multiple residue error correction algorithm can correct up to t residue digit errors with $2t$ redundant moduli by using the base extension method.

Similar to the proposed single residue error correction algorithm, it requires the computation of

$|\Delta|_{m_i}$ for $i = 1, 2, \dots, k, k+1, \dots, k+2t$. $|\Delta|_{m_i}$ can be grouped as follows:

$$\begin{aligned}
S_1 &= \{|\Delta|_{m_{k+1}}, |\Delta|_{m_{k+2}}, \dots, |\Delta|_{m_{k+2t}}\} \\
S_2 &= \{|\Delta|_{m_{k-t}}, |\Delta|_{m_{k-t+1}}, \dots, |\Delta|_{m_{k+t-1}}\} \\
S_3 &= \{|\Delta|_{m_{k-2t-1}}, |\Delta|_{m_{k-2t}}, \dots, |\Delta|_{m_{k-2}}\} \\
&\dots \\
S_{\lceil (k+t-1)/(t+1) \rceil + 1} &= \{|\Delta|_{m_1}, |\Delta|_{m_2}, \dots, |\Delta|_{m_{2t}}\}
\end{aligned} \tag{2.18}$$

The erroneous residue digits can be located by finding S_i with less than or equal to t non-zero elements.

The necessary and sufficient conditions for single residue digit error detection and correction in RRNS were derived in [Bar73]. The minimal redundancy required for single residue digit error correction was determined and a procedure for error correction was proposed. The proposed procedure introduced the m_t -projection concept where the contribution of residue digit in the m_i modulus channel is ignored in computing the magnitude of residue representation. The erroneous residue digit can be located by determining if the computed magnitude of m_t -projection falls in the legitimate range $[0, M_k-1]$. In addition, an algorithm for the correction of single-bit error by using a single redundant modulus was also proposed in [Bar73]. The proposed algorithm, however, is only able to correct certain subclasses of single residue digit error. Later, the same authors presented the arithmetic error detection and correction capabilities of product (AN) codes in RRNS [Bar74]. A product code is formed by multiplying the residue digits with a "generator" A and it is capable of correcting single residue digit error introduced into a legitimate number or an overflow number.

Instead of locating the erroneous residue digits, a different algorithm that utilizes the continued fractions was proposed in [Man76]. The correct error fraction is then used to compute the magnitude of the error-free residue representation. The fraction computations are solved by the recursive Euclidean algorithm. More iterations are required for larger number of moduli and residue digit errors, which make this method inefficient for hardware implementation.

The first practical implementation of error detection and correction in RRNS was presented in [Etz80]. The proposed algorithm is capable of detecting overflow in digital filters in addition to single residue error correction by using the MRC for the m_i -projection method proposed in [Bar73]. According to this paper, overflow and single residue digit error will cause the magnitude of the received residue representation to fall into different illegitimate ranges. Hence, by examining the mixed-radix digits, it can be determined if one of the residue digits is erroneous or the received residue representation overflows. In [Jen83], a new design of error checkers with considerably simple hardware complexity was presented. The error checker contains a pipelined mixed-radix converter for the sequential generation of m_i -projection and two lookup tables for erroneous residue digit correction. In the same paper, a method called expanded projection was proposed to eliminate the need for the alteration of mixed-radix parameters for each m_i -projection using the mixed-radix converter. This method involves the computation of mixed-radix digits of $m_i X_i$ instead of X_i , where X_i is the magnitude of the residue representation in m_i -projection. By using $m_i X_i$, $|X_i|_{m_i}$ is always zero and thus, there is no need to alter the mixed-radix parameters for the m_i -projection. Nonetheless, correcting the residue digit error using this method becomes awkward because the magnitude of the 'true' projections is not computed. The proposed error checkers in [Etz80] and [Jen83] can only detect and correct single residue digit error accurately with the assumption that no fault is introduced during error checking. In order to ensure that the fault introduced into the error checker is detectable, the same author presented a self-checking error checker based on the MRC in [Jen88]. It is claimed that the properties of residue code and residue arithmetic over a set of finite residue rings can be exploited to design self-checking error checker. Essentially, the error introduced into the mixed-radix converter array is equivalent to the class of single residue errors. Thus, it is correctable by using the same technique as that used for single residue digit error correction.

The algorithm proposed in [Ram83] requires a smaller number of iterations than the aforementioned algorithms. In the worst case, the method requires $\lceil 2N/r \rceil + 2$ computations of the magnitude of n residue digits, where N is the total number of moduli and r is the number of redundant moduli. Each iteration involves $N-r$ residue digits to compute the magnitude of the residue representation and check if the magnitude falls in the legitimate range. If the

computed magnitude does not fall into the legitimate range, it will be discarded. Otherwise, it is compared with the magnitudes that are not discarded. The procedure stops when a match is found and the corresponding computed magnitude is the magnitude of the error-free residue representation.

The algorithm proposed in [Su90] enables the implementation of scaling and single residue error correction in the same hardware. The proposed algorithm requires two redundant moduli and their product must be larger than the product of the two largest information moduli. As the error magnitudes of different erroneous residue digits fall in different ranges defined by the two highest order mixed-radix digits of the received residue representation, by using the two highest order mixed-radix digits as the address to a lookup table, the erroneous residue digit can be located and corrected.

The authors of [Kri92] presented a coding theory approach to error control in RRNS and introduced the concepts of Hamming weight, minimum distance, weight distribution, and error detection and correction capabilities in RRNS. Similar to the algorithms in [Wat66] and [Yau73], the algorithm relies on two redundant moduli and the base extension method to compute the test quantities, $|\Delta|_{m_{k+1}}$ and $|\Delta|_{m_{k+2}}$, for single residue digit error correction. From the derivations shown in [Kri92],

$$\begin{aligned} |\Delta|_{m_{k+1}} &\equiv \left| e_i^{(11)} \frac{M_N}{m_i} \right|_{m_{k+1}} \\ |\Delta|_{m_{k+2}} &\equiv \left| e_i^{(12)} \frac{M_N}{m_i} \right|_{m_{k+2}} \end{aligned} \quad (2.19)$$

$$\begin{aligned} |\Delta|_{m_{k+1}} &\equiv \left| \left(e_i^{(21)} - m_i \right) \frac{M_N}{m_i} \right|_{m_{k+1}} \\ |\Delta|_{m_{k+2}} &\equiv \left| \left(e_i^{(22)} - m_i \right) \frac{M_N}{m_i} \right|_{m_{k+2}} \end{aligned} \quad (2.20)$$

where $e_i^{(11)}$, $e_i^{(12)}$, $e_i^{(21)}$ and $e_i^{(22)}$ are the error digits introduced into the modulus channel m_i .

By substituting possible m_i , $e_i^{(11)}$, $e_i^{(12)}$, $e_i^{(21)}$ and $e_i^{(22)}$ can be computed from Equations (2.19) and (2.20). The location of the erroneous residue digit can be determined by checking if $e_i^{(11)} = e_i^{(12)}$ or $e_i^{(21)} = e_i^{(22)}$. If either one of the equations is satisfied, it can be confirmed that the residue digit in the modulus channel m_i is erroneous. The algorithm proposed in [Sun92] is an extension of [Kri92] for multiple residue digit error detection and correction. The number of redundant moduli required is $2t$ for up to t residue digit error correction. The approach taken in [Sun92] is similar to that in [Kri92]. The integers m_i 's in Equations (2.19) and (2.20) become the product of the possible erroneous modulus channels in the multiple residue error detection and correction algorithm.

There are two algorithms proposed in [Ort92] for single residue error correction. Both algorithms use the same concept introduced in [Etz80]. The erroneous residue digit can be located by identifying the range in which the magnitude of received residue representation falls. Instead of computing the complete recombination, the first algorithm uses approximate CRT decoding method to estimate the magnitude of the received residue representation. It requires up to $r + 2$ iterations of detection, where r represents the number of redundant moduli. The second algorithm involves two redundant moduli for overflow consistency check. The erroneous residue digit can be determined by evaluating the overflow count $A(x)$.

The error correction scheme proposed in [Kat96] uses an unconventional approach that does not involve any redundant moduli and the moduli chosen must not be pairwise relatively prime. However, the effect of using moduli that are not coprime is actually the same as the introduction of redundancy [Gar59], which is the essential property for error detection and correction. The proposed scheme relies on the residue property:

$$|x_i|_k = |x_j|_k \quad (2.21)$$

where x_i and x_j are the residue digits in the modulus channels, m_i and m_j , respectively and k is the greatest common divisor of m_i and m_j . The consistency check is carried out by performing Equation (2.21) on all the received residue digits. The erroneous residue digit can be located by

finding the common received residue digit that is involved in all the computations for which Equation (2.21) is not satisfied.

In [Gol00], two algorithms based on CRT, the Unique-Decode and List-Decode, were proposed to detect and correct small and large number of residue errors, respectively. The Unique-Decode method requires the finding of two unknown integers, y and z , of bounded size such that $|y.X|_M \equiv z$, where X is the magnitude of the received residue representation and M is the dynamic range of the RRNS. The finding of y and z has to be carried out recursively until z/y is an integer. The List-Decode method involves forming a polynomial based on the received residue digits. The algorithm searches for a sequence of integers, $d_0, \dots, d_l$, of certain bounded sizes such that $\left| \sum_{i=0}^l d_i r^i \right|_M = 0$ and output all the roots of the polynomial $D(x) = \sum_{i=0}^l d_i x^i$, where l represents the total number of moduli.

The error correction algorithms in [Goh08] are also based on the CRT. Similar to the projection method in [Bar73], the algorithm computes the magnitude of the received residue by ignoring t residue digits at each iteration, where t is the number of correctable residue digit errors. If the computed magnitude of the iteration is within the legitimate range, it can be concluded that the ignored residue digits are erroneous. The maximum number of iterations required in this algorithm is ${}^N C_t$, where N is the total number of moduli. The authors of [Goh08] also proposed an extended scheme to reduce the number of iterations based on the maximum likelihood decoding (MLD) method. The extended scheme involves computing the magnitude of r residue digits, where r is the number of redundant residue digits. When the computed magnitude is within the legitimate range, the remaining residue digits are computed by using the base extension method and then compared with the received residue digits. If the base extended residue digits differ from the received residue digits for t or less locations, it can be concluded that the base extended residue digits are the error-free residue digits. The number of iterations for this extended scheme is $\left\lceil {}^N C_t / {}^r C_t \right\rceil$. This is significantly lower than the first method in [Goh08] especially when the number of moduli is large.

Chapter 3

New Algorithm and Architecture for 2^n Scaling of Signed Integer in $\{2^n-1, 2^n, 2^n+1\}$ RNS

3.1 Introduction

One problem inherited from the non-positional representation of RNS is the truncation of a number in residue domain, which makes overflow avoidance unwieldy. To overcome the magnitude scaling problem, distributed arithmetic filter and number system converters were introduced into an early RNS implementation of Discrete Fourier Transform (DFT) [Tay90]. In [Car04], an RNS implementation of a polyphase filter bank and an 8-point Inverse DFT (IDFT) for digital video broadcasting was shown to have higher speed and lower power consumption than both the error-free and truncated versions of two's complement implementations. To reduce the dynamic range in the filter bank and the IDFT datapath in RNS implementation, truncation was performed on the binary converted intermediate values at the expense of additional latency. The clear advantages of implementing 1-D DWT and 2-D DWT by orthogonal filter banks in RNS were also reported in [Ram03], [Liu04]. From the FPGA implementation results of [Liu04], the throughput is dominated by the reverse converter in each filter bank. To avoid further reverse conversion overhead, the moduli were chosen such that the dynamic range is large enough to obviate overflow. Recently, an adaptive channel equalization filter with a large number of multiply-accumulations was implemented with an allied RNS-binary system to achieve significant power reduction over the TCS implementation without compromising its throughput [Ber07]. Again, scaling was performed in binary via the residue-to-binary conversion to avoid the overflow of dynamic range. In the latest RNS implementation of a 32-bit low-pass finite impulse response (FIR) filter [Maj11], signed number was represented in sign-magnitude form with its magnitude converted to RNS. This atypical RNS representation of signed integer suffers from the dual zero representations and the need to remap the magnitude for regular addition and multiplication, which undermines the advantages

of RNS. From a system perspective, optimizing a subsystem design is often counterproductive unless all the components are fully synergized. If signed number can be scaled in the residue domain without reverting to its two's complement form, all RNS operations for the above applications can be highly optimized in unison to improve the overall accuracy, speed and power consumption.

RNS scaler is usually designed based on either the CRT or the MRC. The scaling factor can be a real number within the dynamic range of the RNS [Gri89], a positive real number which has a magnitude greater than the number of moduli in the moduli set [Ulm98], one of the moduli [Ye08],[Cha11], a product of two or more of the moduli [Su90], [Jul78] or 2^r [Ma10], [Mey03], [Car05], [Low12], where r is a positive integer. Almost all of the existing RNS scalars focus only on magnitude scaling and have either downplayed or glossed over the problem of sign scaling. Because sign detection is itself a difficult operation in RNS, the challenge of implementing a signed over an unsigned integer RNS scaler is the area and delay overheads for sign detection in order to correctly map the scaled signed residues to the legitimate range. As reviewed in Section 2.2 of Chapter 2, the state-of-the-art signed integer RNS scaler [Ye08] comprises an unsigned integer RNS scaler and a correction circuit. The correction circuit involves a dedicated RNS sign detection circuit which is slow and consumes large logic area.

In this chapter, a unified architecture is proposed for the signed integer RNS scaling. The moduli set and scaling factor used are $\{2^n - 1, 2^n, 2^n + 1\}$ and 2^n , respectively. Instead of using a dedicated RNS sign detection circuit, the residue representation of the signed integer scaling result is obtained by manipulating the intermediate computation results. Owing to the parallel and independent residue computation of the signed integer scaling technique and the simplicity of the correction circuit, the proposed architecture is able to achieve a higher computation speed, smaller hardware area and lower power consumption than the latest architecture for the same scaling factor and the same moduli set [Ye08] as well as those designed for different moduli sets of comparable dynamic range [Ma10], [Mey03].

This chapter is organized as follows: In Section 3.2, a simplified circuit for the detection and correction of magnitude scaling error in signed RNS is proposed and the detailed implementation of the proposed signed integer RNS scaler is presented. The complexity analysis and comparison of different signed scaler implementations based on unit-gate model, logic and physical synthesis results are made in Section 3.3. Finally, the chapter is summarised in Section 3.4.

3.2 Proposed Scaling Algorithm and Architecture

3.2.1 Magnitude Scaling Error in Signed RNS

Let $Y \equiv (y_1, y_2, y_3)$ be the residues obtained by scaling $X \equiv (x_1, x_2, x_3)$ by k in the residue domain of RNS $\{2^n-1, 2^n, 2^n+1\}$. For $m_1 = 2^n-1$, $m_2 = 2^n$ and $m_3 = 2^n+1$, $|M_1^{-1}|_{m_1} = 2^{n-1}$, $|M_2^{-1}|_{m_2} = 2^{n-1}$ and $|M_3^{-1}|_{m_3} = 2^{n-1}+1$ [Con99], where $M_i = M/m_i$ and $|M_i^{-1}|_{m_i}$ is the multiplicative inverse of $|M_i|_{m_i}$. According to the CRT [Omo07], [Moh02] given by Equation (2.14) of Chapter 2, the integer Y can be recovered by:

$$Y = \left| \sum_{i=1}^N M_i \left| |M_i^{-1}|_{m_i} y_i \right|_{m_i} \right|_M \quad (3.1)$$

$$Y = \left| 2^n (2^n + 1) 2^{n-1} y_1 + (2^n - 1)(2^n + 1)(2^n - 1) y_2 + (2^n - 1) 2^n (2^{n-1} + 1) y_3 \right|_{(2^{2n} - 1) 2^n}$$

However, if X is treated as a signed integer $\tilde{X}$ and represented in signed RNS, Equation (3.1) may not yield the correct result of scaling X by k . This is because the scaling of magnitude in unsigned RNS creates an ambiguity in the range partitioning of signed RNS, causing the scaled residues to be incorrectly mapped for negative integers.

The range mismatch can be resolved upon detecting the sign of the input operand. Unfortunately, the sign of an integer is not distinguishable from its residue representation. Existing solutions to the RNS sign detection problem are mainly based on the CRT, the MRC or the core function [Tom08], [Hen99], [Abt05]. ROM matrices are required to find the orthogonal projections in these approaches, making the overall circuit in conjunction with the magnitude

scaling module to be more expensive than a residue-to-binary converter. To significantly lower the cost of a signed integer scaling circuit, the sign detection and correction circuit must be kept simple and coherent with the magnitude scaling module.

3.2.2 Correction Circuit for Signed RNS Scaling

From the definition of scaling in RNS, the following expressions of the scaled residues (y_1 , y_2 , y_3) in terms of (x_1 , x_2 , x_3) of an unsigned integer X can be derived [Cha11], [Moh02]:

$$y_1 = |x_1 - x_2|_{m_1} \quad (3.2)$$

$$y_2 = \left\| \left((2^{2n-1} + 2^{n-1})x_1 - 2^n x_2 + (2^{2n-1} + 2^{n-1} - 1)x_3 \right) \right\|_{m_1 m_3} \Big|_{m_2} \quad (3.3)$$

$$y_3 = |x_2 + 2^n x_3|_{m_3} \quad (3.4)$$

where $k = 2^n$.

In order to obtain the correct residue representation of $\tilde{Y}$, y_1 , y_2 and y_3 need to be corrected by detecting the sign of $\tilde{X}$ to map X to $\tilde{X}$ based on their relationship expressed in Equation (2.4). No correction is required when $\tilde{X} \geq 0$ because $\tilde{X} = X$ according to Equation (2.4). Hence, $\tilde{y}_1 = y_1$, $\tilde{y}_2 = y_2$ and $\tilde{y}_3 = y_3$.

However, when $\tilde{X} < 0$, $\tilde{X} = X - M$. Since $k = m_2$,

$$\tilde{Y} = \left\lfloor \frac{\tilde{X}}{k} \right\rfloor = \left\lfloor \frac{X - M}{k} \right\rfloor = \left\lfloor \frac{X}{k} - m_1 m_3 \right\rfloor \quad (3.5)$$

Since $m_1 m_3$ is an integer, Equation (3.5) can be rewritten as:

$$\tilde{Y} = \left\lfloor \frac{X}{k} \right\rfloor - m_1 m_3 = Y - m_1 m_3 \quad (3.6)$$

The residue representation of $\tilde{Y}$ when $\tilde{X} < 0$ can be computed as follows [Ye08]:

$$\tilde{y}_1 = |Y - m_1 m_3|_{m_1} = \left| |Y|_{m_1} - |m_1 m_3|_{m_1} \right|_{m_1} = |y_1 - 0|_{m_1} = y_1 \quad (3.7)$$

$$\tilde{y}_2 = |Y - m_1 m_3|_{m_2} = \left| |Y|_{m_2} - |m_1 m_3|_{m_2} \right|_{m_2} = |y_2 + 1|_{m_2} \quad (3.8)$$

$$\tilde{y}_3 = |Y - m_1 m_3|_{m_3} = \left| |Y|_{m_3} - |m_1 m_3|_{m_3} \right|_{m_3} = |y_3 - 0|_{m_3} = y_3 \quad (3.9)$$

From Equations (3.7) and (3.9), no correction is needed for $\tilde{y}_1$ and $\tilde{y}_3$ when $\tilde{X} < 0$, but y_2 needs to be incremented by one to obtain $\tilde{y}_2$ as indicated in Equation (3.8). To detect $\tilde{X} < 0$ from its residues, a full sign detection circuit is required. This is to be avoided because a full sign detection circuit is expensive to be implemented on hardware.

Since the dynamic range $M = 2^{3n} - 2^n$ of the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is always even, the residue representation of X can be mapped to that of $\tilde{X}$ for X in the range of $[0, M/2 - 1]$ if $\tilde{X} \geq 0$ and $[M/2, M - 1]$ if $\tilde{X} < 0$. In order to correct Y in an unsigned RNS to $\tilde{Y}$ in a signed RNS, it is necessary to know when $\tilde{Y}$ becomes negative. The ranges of the scaled unsigned integer Y for positive and negative $\tilde{X}$ are determined as follows:

Since $Y = \left\lfloor \frac{X}{k} \right\rfloor$, when $\tilde{X} \geq 0$,

$$\begin{aligned} 0 &\leq X \leq \frac{M}{2} - 1 \\ 0 &\leq Y \leq \left\lfloor \frac{M - 2}{2k} \right\rfloor \\ 0 &\leq Y \leq 2^{2n-1} - 1 \end{aligned} \quad (3.10)$$

When $\tilde{X} < 0$,

$$\begin{aligned} \frac{M}{2} &\leq X \leq M - 1 \\ \left\lfloor \frac{M}{2k} \right\rfloor &\leq Y \leq \left\lfloor \frac{M - 1}{k} \right\rfloor \\ 2^{2n-1} - 1 &\leq Y \leq 2^{2n} - 2 \end{aligned} \quad (3.11)$$

From Equations (3.10) and (3.11), when $Y > 2^{2n-1} - 1$, $\tilde{Y}$ is negative and when $Y < 2^{2n-1} - 1$, $\tilde{Y}$ is positive. Thus, the condition when $\tilde{Y}$ is negative can be detected by the $(2n-1)$ -th bit of Y , i.e., $(Y)_{2n-1}$ being '1'. According to Equation (3.8), this bit can be added at the least significant bit

(LSB) of y_2 to correct the sign error of the magnitude-scaled residue, i.e., $\tilde{y}_2 = |y_2 + (Y)_{2n-1}|_{2^n}$.

By adopting [Cha11] for magnitude scaling, no additional circuit is needed to determine $(Y)_{2n-1}$ because Y is available in the result of Equation (3.3) before taking the modulo of m_2 .

However, when $Y = 2^{2n-1} - 1$, $\tilde{Y}$ can be either positive or negative according to Equations (3.10) and (3.11). Based on the definition of magnitude scaling, $Y = \lfloor X/2^n \rfloor$, the range of X for $Y = 2^{2n-1} - 1$ is given by:

$$\begin{aligned} 2^n(2^{2n-1} - 1) &\leq X \leq 2^n(2^{2n-1} - 1) + 2^n - 1 \\ 2^{3n-1} - 2^n &\leq X \leq 2^{3n-1} - 1 \end{aligned} \quad (3.12)$$

From Equation (3.12), there are altogether 2^n integers in $[2^{3n-1} - 2^n, 2^{3n-1} - 1]$. From Equation (2.4), when $X \geq M/2 = 2^{3n-1} - 2^{n-1}$, $\tilde{X} < 0$. Therefore, for $Y = 2^{2n-1} - 1$, when $\tilde{X} \geq 0$ and $\tilde{Y} \geq 0$,

$$2^{3n-1} - 2^n \leq X \leq 2^{3n-1} - 2^{n-1} - 1 \quad (3.13)$$

and when $\tilde{X} < 0$ and $\tilde{Y} < 0$,

$$2^{3n-1} - 2^{n-1} \leq X \leq 2^{3n-1} - 1 \quad (3.14)$$

Since the magnitude of X is not directly available at the input, it can only be determined from the residue representation. By taking the modulo 2^n operation on Equation (3.14), the range of the residue x_2 is found to be $2^{n-1} \leq x_2 \leq 2^n - 1$. Since $\tilde{x}_2 \equiv x_2$, $2^{n-1} \leq \tilde{x}_2 \leq 2^n - 1$. Let $(x_i)_j$ denote the j -th bit of x_i . It is observed that the bit $(\tilde{x}_2)_{n-1}$ is always '1' for $\tilde{x}_2$ in the range of Equation (3.14) and '0' otherwise. Hence, when $Y = 2^{2n-1} - 1$, $(\tilde{x}_2)_{n-1}$ alone is sufficient to determine the sign of $\tilde{Y}$, i.e., $(\tilde{x}_2)_{n-1}$ is '1' when $\tilde{Y} < 0$ and '0' when $\tilde{Y} \geq 0$.

In the above discussion, it is proven that the sign of $\tilde{Y}$ can be simply detected by using the bit $(Y)_{2n-1}$ except for the case when $Y = 2^{2n-1} - 1$ where an extra bit $(\tilde{x}_2)_{n-1}$ is needed to determine the sign of $\tilde{Y}$. The condition of $Y = 2^{2n-1} - 1$ can be determined by checking if $2n-1$ LSBs of Y are '1's and the most significant bit (MSB) of Y is '0'. However, this implementation results in a high fan-in multi-level logic structure, which is very slow and costly when n is large. It can be

shown in the followings that the fan-in of the multi-level logic gate circuit for the detection of the condition of $Y = 2^{2n-1}-1$ can be reduced from $2n$ bits to only $n+2$ bits, namely the n -bit residue y_1 , and two other bits, $(Y)_{2n-1}$ and $(y_2)_{n-1}$.

When $Y = 2^{2n-1}-1$, $(Y)_{2n-1} = '0'$ and

$$y_1 = \left| 2^{2n-1} - 1 \right|_{2^n-1} = \left| (2^n)(2^{n-1}) - 1 \right|_{2^n-1} = 2^{n-1} - 1 \quad (3.15)$$

From Equation (3.15), y_1 can be computed as $2^{n-1}-1$ when $Y = 2^{2n-1}-1$. However, y_1 alone is not sufficient to determine if $Y = 2^{2n-1}-1$ because there are 2^n+1 different integers of Y that have $y_1 = 2^{n-1}-1$ in their residue representation. Among these integers, $2^{n-1}+1$ of them have values less than or equal to $2^{2n-1}-1$. When $y_1 = 2^{n-1}-1$ and $Y \leq 2^{2n-1}-1$, Y has the values of $2^{n-1}-1$, $2^{n-1}+2^n-2$, ..., $2^{2n-1}-2^n$ and $2^{2n-1}-1$. Thus, $y_2 = |Y|_{2^n} = 2^{n-1}-1, 2^{n-1}-2, \dots, 0, 2^n-1$. Among these y_2 values, only when $Y = 2^{2n-1}-1$ can y_2 have value greater than $2^{n-1}-1$ (i.e., $(y_2)_{n-1} = '1'$). In other words, $(Y)_{2n-1}$, y_1 and $(y_2)_{n-1}$ are sufficient to determine if $Y = 2^{2n-1}-1$. Figure 3.1 depicts the proposed architecture of the signed integer RNS scaler. The architecture of unsigned integer RNS scaler [Cha11] is enclosed in the dashed-line box and the correction circuit is enclosed in the solid-line box. The role of the control signal generation in the correction circuit is to detect the condition when $Y = 2^{2n-1}-1$ and $\tilde{Y}$ is in negative range. Under this condition, '1' will be selected as the LSB of the correction factor. Otherwise, $(Y)_{2n-1}$ will be selected. The control signal generation circuit can be built using $n+2$ two-input *AND* gates with y_1 , $(\tilde{x}_2)_{n-1}$, $(Y)_{2n-1}$ and $(y_2)_{n-1}$ as the inputs.

Figure 3.1 Proposed two-stage signed integer RNS scaler.

3.2.3 Unified Signed Integer RNS Scaler Architecture

The architecture depicted in Figure 3.1 is a direct implementation based on the mathematical manipulation discussed earlier. The area and speed can be further improved by merging the two stages of computations into one by eliminating some redundancies in the arithmetic circuits. The single stage architecture is shown in Figure 3.2. The modulo 2^{2n-1} adder of the magnitude scaler and correction circuit of Figure 3.1 are integrated into a “merged sign detection and correction” module. It comprises a modified mod 2^n adder with c_{in} , a simplified mod 2^n adder, a modified control signal generation block and an *AND* gate array as shown in Figure 3.2.

According to [Cha11],

$$\left\| A + B \right\|_{2^{2n}-1} \Big|_{2^n} = \begin{cases} \left\| A + B \right\|_{2^n} + 1 \Big|_{2^n} & \text{if } A + B \geq 2^{2n} \\ \left\| A + B \right\|_{2^n} & \text{otherwise} \end{cases} \quad (3.16)$$

From Equation (3.16), $\left\| A + B \right\|_{2^{2n}-1} \Big|_{2^n}$ can be implemented by a simple mod 2^n adder provided that when $A+B \geq 2^{2n}$, the sum is incremented by one. For simplicity, this simplified implementation of Equation (3.16) is succinctly denoted by:

$$\left\| A + B \right\|_{2^{2n}-1} \Big|_{2^n} = \left\| A + B + c_{in} \right\|_{2^n} \quad (3.17)$$

where $c_{in} \in \{0,1\}$ is the carry input to the modified mod 2^n adder with c_{in} shown in Figure 3.2.

Figure 3.2 Proposed simplified signed integer RNS scaler.

Figure 3.3(a) depicts an example of the proposed mod 2^n adder with c_{in} for $n = 4$. The computation of y_2 in Equation (3.3) can be performed by using the proposed modified mod 2^n adder with c_{in} . This adder adds two n -bit operands taken from the n LSBs of the sum and carry vectors, A and B , produced by the $2n$ -bit CSA with EAC block shown in Figure 3.2. This adder has a similar structure as a standard mod 2^n adder with an additional prefix level. This extra level of prefix operators is enclosed in the dashed-line box of Figure 3.3(a). They are used to generate the carry signals due to c_{in} [Zim99]. The signal, c_{in} in this case is equal to c_{2n-1} , which is the carry output generated from the $A+B$ operation. By using this adder, the use of a large and slow mod 2^{2n-1} adder can be avoided.

Besides being one of the inputs to the modified control signal generation circuit, $(Y)_{2n-1}$ is also the LSB of the correction factor to map y_2 to $\tilde{y}_2$. The mapping is done by using a simplified mod 2^n adder which has a structure shown in Figure 3.3(b), where $(Y)_{2n-1}$ is added to y_2 to obtain $\tilde{y}_2'$, i.e., $\tilde{y}_2' = |y_2 + Y_{2n-1}|_{2^n}$. Since all bits except the LSB of the correction factor are '0', the prefix operator, denoted by ' $\bullet$ ', is simplified to ' $\odot$ ' to reduce the critical path delay.

Figure 3.4 shows the circuit of the modified control signal generation block which is similar to that in Figure 3.1 with the addition of a $(Y)_{2n-1}$ generation block. The role of the $(Y)_{2n-1}$ generation block is to compute $(Y)_{2n-1}$ and c_{2n-1} bits based on the n MSBs of A and B , and G_{n-1}

and P_{n-1} from the modified mod 2^n adder with c_{in} . The c_{2n-1} bit is fed as the carry input c_{in} to the modified mod 2^n adder with c_{in} shown in Figure 3.3(a).

In Figure 3.1, a multiplexer is needed to select the correction factor to be added to y_2 . As $(y_2)_{n-1}$ is an input to the control signal generation block, addition of the correction factor to y_2 is delayed. In Figure 3.2, $(Y)_{2n-1}$ is added to y_2 before the corrected output is selected by an *AND* gate array. This efficient implementation is derived as follows.

When $Y = 2^{2n-1}-1$ and $\tilde{Y}$ is in negative range,

$$\tilde{y}_2 = |y_2 + 1|_{2^n} = |2^n - 1 + 1|_{2^n} = |2^n|_{2^n} = 0 \quad (3.18)$$

For other conditions, $(Y)_{2n-1}$ can just be added to y_2 ,

$$\tilde{y}_2 = |y_2 + (Y)_{2n-1}|_{2^n} \quad (3.19)$$

From Equations (3.18) and (3.19), the corrected output is either 0 or $|y_2 + (Y)_{2n-1}|_{2^n}$. Instead of using n -way multiplexer, n two-input *AND* gates will suffice to obtain the correct output.

Figure 3.3 (a) Modified mod 2^n adder with c_{in} for $n = 4$ (b) Simplified mod 2^n adder for $n = 4$.

Figure 3.4 Modified control signal generation module.

3.3 Evaluation and Comparison

In this section, the area and time complexities of the proposed architecture depicted in Figure 3.2 are first analyzed based on the unit-gate model [Ver02] before they are synthesized, optimized and physically implemented in an integrated circuit design flow to evaluate its hardware area, delay and power consumption. The results are compared against the latest signed integer RNS scaler architectures proposed in [Ye08], [Mey03] and [Ma10] for four different dynamic ranges of 15, 18, 21 and 24 binary bits. The three moduli sets $\{2^n-1, 2^n, 2^n+1\}$ used for the proposed design and [Ma10] are $\{31, 32, 33\}$, $\{63, 64, 65\}$, $\{127, 128, 129\}$ and $\{255, 256, 257\}$ with exact dynamic ranges of 32,736, 262,080, 2,097,024 and 16,776,960 for $n = 5, 6, 7$ and 8, respectively. These moduli sets are inapplicable for the architectures of [Mey03] and [Ma10] as all its moduli must be odd integers. For a fair comparison, the moduli sets $\{23, 29, 31\}$, $\{53, 59, 61\}$, $\{109, 113, 127\}$ and $\{239, 241, 251\}$ with comparable dynamic ranges (DRs) of 15, 18, 21 and 24 bits and the same scaling factor of $n = 5, 6, 7$ and 8, respectively are used for [Mey03] and [Ma10]. These moduli sets are made up of adjacent prime numbers and are chosen in favor of [Mey03] and [Ma10] such that their exact dynamic ranges, which are 20,677, 190,747, 1,564,259 and 14,457,349 just fall below those of $\{2^n-1, 2^n, 2^n+1\}$ for $n = 5, 6, 7$ and 8, respectively.

3.3.1 Unit-gate Analysis

Unit-gate model [Ver02] is adopted for the first order estimation of the hardware area and time complexities. Under this analytical model, a two-input monotonic gate, such as *AND*, *OR*,

NAND or *NOR* gate, has one unit of area and one unit of delay, an *XOR* gate has two units of area and two units of delay, and an inverter is assumed to have zero unit of area and delay [Ver02].

The unit-gate areas of the arithmetic blocks that compute $\tilde{y}_1$, $\tilde{y}_2$ and $\tilde{y}_3$ for the proposed architecture are evaluated independently. Channels m_1 and m_3 have similar architectures as those proposed in [Cha11] and their areas excerpted from Table III of [Cha11] are $3n\lceil\log_2 n - 1\rceil + 12n$ and $4.5n\lceil\log_2 n\rceil + 7.5n + 12$ units, respectively. Channel m_2 consists of bit rewiring, $2n$ -bit end-around-carry carry save adder ($2n$ -bit EAC CSA) and merged sign detection and correction blocks. The bit rewiring and $2n$ -bit EAC CSA blocks are built up of n *OR* gates and $2n$ full adders (FAs). Since a FA consumes 7 units of area, these two blocks have a total unit-gate area of $15n$. The unit-gate area for the merged sign detection and correction module does not have a simple closed form expression of n . The unit-gate areas of its constituent components for $n = 5, 6, 7$ and 8 are listed in Table 3.1, where A_{MMA} , A_{CS} , A_{SMA} and A_{AND} denote the unit-gate areas of the modified mod 2^n adder with c_{in} , control signal generation block, simplified mod 2^n adder and *AND* gate array, respectively. The unit-gate areas of all other modules are added together under A in the ‘Others’ column. All component areas are summed to obtain the overall unit-gate area of the proposed design.

The unit gate delays of channels m_1 and m_3 of the proposed architecture are $2\lceil\log_2 n - 1\rceil + 3$ and $2\lceil\log_2 n\rceil + 9$ units, respectively, as excerpted from Table IV of [Cha11]. In channel m_2 , the total unit-gate delay of the bit rewiring and $2n$ -bit EAC CSA blocks is 5 units. Meanwhile, the longest path of merged sign detection and correction module goes from the control signal generation block to the *AND* gate array through the simplified mod $2n$ adder. The unit-gate delay of the control signal generation block works out to be $2\lceil\log_2 n\rceil + 7$. A simple closed form expression of n cannot be derived for the unit-gate delay of the simplified mod $2n$ adder and it will be evaluated individually for $DR = 15, 18, 21$ and 24 . Channel m_2 has the longest path delay among the three parallel channels and it determines the worst case delay of the proposed architecture. Its unit-gate delay is shown in Table 3.1, where D_{CS} , D_{SMA} and D_{AND}

denote the unit-gate delays of the control signal generation block, simplified mod $2n$ adder and AND gate array, respectively. The unit-gate delays of the bit-rewiring and $2n$ -bit EAC CSA blocks are consolidated into D in the ‘Others’ column of Table 3.1.

Table 3.1 Unit-gate Area and Delay of Proposed Design

n	Merged sign detection & correction							Others		Overall	
	A_{MMA}	A_{CS}	A_{SMA}	A_{AND}	D_{CS}	D_{SMA}	D_{AND}	A	D	A	D
5	57	45	21	5	13	6	1	282	5	410	25
6	72	51	25	6	13	7	1	336	5	490	26
7	87	57	32	7	13	8	1	390	5	573	27
8	105	66	39	8	13	8	1	444	5	662	27

As the detailed implementations of mod 2^n+1 subtractor and mod 2^n-1 subtractor for the computations of Equations (3.2) and (3.4), respectively in Fig. 1 of [Ye08] are not provided in [Ye08], it is assumed that they consume the same unit-gate area and delay as those datapath implementations of channel m_1 and m_3 of the proposed architecture. In addition, the architecture of [Ye08] requires two mod 2^n adders, n 2-to-1 multiplexers and an RNS sign detection module. These modular adders and multiplexers are also assumed to have the same implementations as those in the proposed architecture. The RNS sign detection module in [Ye08] is implemented based on the architecture suggested in [Tom08], which consumes $19n-4$ units of area. The total unit-gate area of the architecture of [Ye08] adds up to $13.5n\lceil\log_2 n\rceil + 56.5n + 8$ units, as shown in Table 3.2.

Table 3.2 Unit-gate Area of [Ye08]

Module	Area per Module	#	Total Module Area
mod 2^n-1 adder	$3n\lceil\log_2 n-1\rceil + 12n$	2	$6n\lceil\log_2 n-1\rceil + 24n$
mod 2^n adder	$1.5n\lceil\log_2 n\rceil + 5n$	2	$3n\lceil\log_2 n\rceil + 10n$
mod 2^n+1 adder	$4.5n\lceil\log_2 n\rceil + 7.5n + 12$	1	$4.5n\lceil\log_2 n\rceil + 7.5n + 12$
MUX	$2n$	1	$2n$
Sign Detection	$19n-4$	1	$19n-4$
Total Area			$13.5n\lceil\log_2 n\rceil + 56.5n + 8$

From Fig. 1 of [Ye08], the critical path delay of [Ye08] is constituted by the delays of a mod 2^n+1 subtractor, a mod 2^n-1 subtractor, two mod 2^n adders and a 2-to-1 multiplexer. Based on the earlier assumption about the implementations of mod 2^n+1 and mod 2^n-1 subtractors, the unit-gate delay of [Ye08] is analyzed to be $8\lceil \log_2 n \rceil + 18$ units.

In [Mey03], the arithmetic operations are implemented by lookup tables (LUTs). These LUTs are usually implemented with ROM modules. According to the ROM model of [Ulm98], the number of transistors T_{ROM} and unit-gate delay D_{ROM} of a $2^n \times p$ ROM can be approximated by

$$T_{\text{ROM}(2^n \times p)} = 2^{\lceil n/2 \rceil} (\lceil n/2 \rceil + 1) + 2^n p + 2^{\lfloor n/2 \rfloor} p (\lfloor n/2 \rfloor + 2) + p(2^{\lfloor p/2 \rfloor} + 1) \quad (3.20)$$

$$D_{\text{ROM}(2^n \times p)} = (1 + \lceil \log_2 n \rceil + \lceil n/2 \rceil) D_{\text{NAND}} \quad (3.21)$$

where n is the address bit-width and p is the data bit-width.

For ease of comparison, the areas of logic gates are also expressed in transistor count. To convert the number of unit gates into the equivalent number of transistors, static CMOS logic style is assumed, where a FA is typically implemented with 28 transistors [Cha11]. This makes one unit-gate area equivalent to four transistors.

The design of [Mey03] consists of n scaling-by-2 modules and a sign detection module. Each scaling-by-2 module is composed of a modular arithmetic module and a parity detection module. The implementations of the sign detection and parity detection modules were not described in [Mey03]. According to [Car05], where a similar sign integer RNS scaler as [Mey03] is proposed, they are assumed to be based on the architectures of [Vu85] and [She89b], respectively.

As suggested by [Car05], the scaling-by-2 module in [Mey03] can be implemented with an additional redundant modulus m_r . To minimize the overhead due to the redundant modulus, $m_r = 3$ is chosen. The modular arithmetic module that performs modular addition and scaling-by-2 operations consists of six $2^n \times n$ ROMs, three $2^2 \times 2$ ROMs, $3n+4$ 2-to-1 multiplexers and an XOR gate. Based on the design proposed in [She89b], the parity detection module used in [Mey03]

consists of three $2^n \times 2$ ROMs, a $2^2 \times 2$ ROM, two $2^4 \times 2$ ROMs, a $2^4 \times 1$ ROM, three $2^n \times 1$ ROMs and three XOR gates. The numbers of transistors T_A and T_{PD} for the arithmetic and parity detection modules, respectively are shown in Table 3.3. Their sum gives the estimated total number of transistors T_{S1} required for the scaling-by-2 module.

Based on the design proposed in [Vu85], the sign detection module for [Mey03] consists of three $2^n \times p$ ROMs, a p -bit carry save adder and a p -bit simplified parallel prefix adder, where $p = 17, 21, 24$ and 27 for $n = 5, 6, 7$ and 8 , respectively. The p -bit carry save adder consists of p FAs and its number of transistors T_{CSA} is approximated to $28p$. The p -bit simplified parallel prefix adder is used to compute the MSB of the sum of two p -bit operands for the sign detection module. The number of transistors T_{SPPA} of the p -bit simplified parallel prefix adder is individually estimated for each DR in Table 3.3. The total number of transistors T_{S2} of a sign detection module is the sum of T_{ROM} , T_{CSA} and T_{SPPA} . Overall, the estimated total number of transistors required for the signed scaler of [Mey03] in Table 3.3 is calculated by $nT_{S1} + T_{S2}$ for each DR .

Table 3.3 Estimated Number of Transistors of [Mey03]

n	Scaling-by-2			Sign detection				Overall
	T_{PD}	T_A	T_{S1}	T_{ROM}	T_{CSA}	T_{SPPA}	T_{S2}	
5	912	2,032	2,944	15,651	476	364	16,491	31,211
6	1,416	4,534	5,950	71,223	588	452	72,263	107,963
7	2,280	8,212	10,492	307,320	672	524	308,516	381,960
8	3,936	18,514	22,450	692,385	756	584	693,725	873,325

The delay D_{SD} of the sign detection module in [Mey03] is contributed by the aggregated delay of a $2^n \times p$ ROM, a FA, and a p -bit simplified parallel prefix adder, where $p = 17, 21, 24$ and 27 for $n = 5, 6, 7$ and 8 , respectively. The delay D_{PD} of the parity detection module is contributed by the aggregated delay of a $2^n \times 2$ ROM, a $2^4 \times 2$ ROM, a $2^4 \times 1$ ROM, and a XOR gate. The sign detection is performed only once and it affects only the delay of the first scaling-by-2 module. The critical path of the first scaling-by-2 module D_{S1} is formed from the sign detection module to the $2^2 \times 2$ ROM in the m_r channel while the critical path of the subsequent scaling-by-2 modules D_{S2} is formed from the parity detection module to the $2^2 \times 2$ ROM in the m_r channel.

Therefore, the overall delay is calculated by $D_{S1} + (n-1)D_{S2}$. Table 3.4 shows the breakdown of unit-gate delays of [Mey03] with scaling factor of 2^n for $n = 5$ to 8.

Table 3.4 Unit-gate Delay of [Mey03]

n	First scaling-by-2			Other scaling-by-2			Overall
	D_{SD}	D_{others}	D_{S1}	D_{PD}	D_{others}	D_{S2}	
5	21	12	33	19	12	31	157
6	23	12	35	19	12	31	190
7	24	12	36	20	12	32	228
8	24	12	36	20	12	32	260

The design of [Ma10] consists of an n LSB conversion, a sign detection and an arithmetic modules. As suggested by [Ma10], the n LSB conversion module is implemented based on the architecture of [She89b]. $m_r = 3$ is chosen for the additional redundant modulus as in the implementation of [Ma10]. The n LSB conversion module consists of three $2^n \times 2$ ROMs, a $2^2 \times 2$ ROM, two $2^4 \times 2$ ROMs, a $2^4 \times n$ ROM, three $2^n \times n$ ROMs, two n -bit carry save adders and a mod 2^n adder. The transistor count of the mod 2^n adder is estimated to be $6n \lceil \log_2 n \rceil + 20n$. Its total transistor count T_{conv} is recorded in Table 3.5. According to [Ma10], the sign of X can be detected by finding the parity of $Z = 2X$ by using

$$sign(X) = parity(Z) = LSB\left(\left\|M_1^{-1}\right\|_{m_1} z_1\right) \oplus \dots \oplus LSB\left(\left\|M_N^{-1}\right\|_{m_N} z_N\right) \oplus LSB(r_z) \quad (3.22)$$

where $LSB(x)$ represents the LSB of x and $\oplus$ denotes the XOR operation. The algorithm in [She89b] is suggested by [Ma10] for the computation of r_z but this will produce an incorrect answer. This is because when $X < 0$, Z will exceed the dynamic range of the moduli set and $z_r = \left\|Z\right\|_{m_r}$ is no longer equal to $\left\|2 \times X\right\|_{m_r}$. For a fair comparison, the sign detection module of [Ma10] is implemented in the same way as that of [Mey03] and its transistor count T_{SD} in Table 3.5 is the same as T_{S2} in Table 3.3. The arithmetic module of [Ma10] consists of three $2^{2n} \times n$ ROMs, six $2^n \times n$ ROMs and $3n$ 2-to-1 multiplexers. Its transistor count T_A is shown in Table 3.5.

The delay D_{conv} of the n LSB conversion module of [Ma10] is contributed by the aggregated delay of a $2^n \times 2$ ROM, a $2^4 \times 2$ ROM, a $2^4 \times 6$ ROM, a FA, and a mod 2^n adder. The unit-gate

delay of mod 2^n adder is $2\lceil \log_2 n \rceil + 3$. The delay D_A of the arithmetic module of [Ma10] is contributed by the aggregated delay of a $2^{2n} \times n$ ROM, two $2^n \times n$ ROMs and a 2-to-1 multiplexer. The longest path of [Ma10] is formed from the n LSB conversion module to the arithmetic module. Thus, the unit-gate delay of [Ma10] is obtained by the sum of D_{conv} and D_A , as shown in Table 3.5.

Table 3.5 Number of Transistors and Unit-gate Delay of [Ma10]

n	Number of Transistors			Overall	Delay		Overall
	T_{conv}	T_{SD}	T_A		D_{conv}	D_A	
5	2,154	16,491	21,177	39,822	30	26	56
6	3,884	72,263	88,734	164,881	30	27	57
7	6,374	308,516	379,455	694,345	31	30	61
8	12,816	693,725	1,659,840	2,366,381	31	31	62

Table 3.6 compares the transistor counts and unit-gate delays of the proposed design, [Ye08], [Mey03] and [Ma10] for four dynamic ranges. It shows that the transistor counts of the proposed architecture are 16.84%, 16.95%, 16.59% and 15.56% lesser than [Ye08], 94.74%, 98.18%, 99.40% and 99.70% lesser than [Mey03], and 95.88%, 98.81%, 99.67% and 99.88% lesser than [Ma10] for DR = 15, 18, 21 and 24 bits, respectively. The proposed architecture is also 40.48%, 38.10%, 35.71% and 35.71% faster than [Ye08], 84.08%, 86.32%, 88.16% and 89.62% faster than [Mey03], and 55.36%, 54.39%, 55.74% and 56.45% faster than [Ma10] for DR = 15, 18, 21 and 24 bits, respectively.

Table 3.6 Comparison of Number of Transistors and Unit-gate Delay (Value in Parenthesis Indicates the Percentage Reduction of Transistor Count or Delay of Proposed Design Over Its Contender)

DR (bit)	Number of Transistors				Unit-gate Delay			
	This	[Ye08]	[Mey03]	[Ma10]	This	[Ye08]	[Mey03]	[Ma10]
15	1,640	1,972 (16.84)	31,211 (94.74)	39,822 (95.88)	25	42 (40.48)	157 (84.08)	56 (55.36)
18	1,960	2,360 (16.95)	107,963 (98.18)	164,881 (98.81)	26	42 (38.10)	190 (86.32)	57 (54.39)
21	2,292	2,748 (16.59)	381,960 (99.40)	694,345 (99.67)	27	42 (35.71)	228 (88.16)	61 (55.74)
24	2,648	3,136 (15.56)	873,325 (99.70)	2,366,381 (99.88)	27	42 (35.71)	260 (89.62)	62 (56.45)

3.3.2 Logic Synthesis Results

To account for the fanout and logic optimization effects on different designs and to enable the comparison of their power consumptions, each design is specified at gate level in Verilog HDL and implemented by a standard cell design flow. The designs are functionally verified using ModelSim, and synthesized and technology mapped to TSMC 0.18 μm CMOS technology standard cell library using Synopsys Design Compiler version Y-2006.06-SP4. The operating conditions are set to the worst case conditions where the operating supply voltage and temperature are 1.62V and 125°C, respectively. Standard buffers of strength 2X are used for both input drive and output load. Each design is independently and recursively optimized for speed to obtain their minimum achievable delay, which determines its maximum operating frequency.

The areas and delays of these designs after logic synthesis and optimization are shown in Table 3.7. The results show that the areas of the proposed design are 21.59%, 48.06%, 51.25% and 36.61% smaller than those of [Ye08], 87.36%, 93.95%, 95.84% and 96.79% smaller than those of [Mey03], and 89.57%, 96.70%, 98.44% and 99.36% smaller than those of [Ma10] for DR = 15, 18, 21 and 24 bits, respectively. The comparison with [Ye08] in terms of synthesized area suggests that more aggressive area savings can be obtained from the actual implementation of the design than the theoretical estimation based on transistor count. Comparing with [Ye08], the proposed design is 28.83%, 35.76%, 35.89% and 40.12% faster for DR = 15, 18, 21 and 24 bits, respectively. The delay improvement estimated by the unit-gate analysis is somewhat optimistic for the lower dynamic range, but is fairly accurate otherwise. Comparing with [Mey03] and [Ma10], the proposed design speeds up by more than 84.52% and 63.10%, respectively. The speed improvements obtained by the unit-gate delay model are relatively well matched for [Mey03] but underestimated for [Ma10].

As the dynamic power dissipation increases linearly with frequency and quadratically with supply voltage, the power consumption of all circuits are measured using Synopsys PrimeTime PX at the same clock rate and the same supply voltage of 1.62V. For each dynamic range, a common clock period is set based on the slowest design of Table 3.7. As the dynamic power of

a combinational circuit is dependent on the input patterns, a Monte Carlo simulation method [Bur93] using a finite number of randomly generated test patterns is adopted to estimate the average power dissipation. In this power simulation, 500 random input vectors were injected with 99% confidence that the estimation error is bounded below 2.5%. The simulated leakage power and total power dissipation are listed in Table 3.8, where the total power consists of average dynamic power and leakage power. The proposed design saves more than 32% of power over [Ye08], more than 97% over [Mey03], and more than 89% over [Ma10], despite operating at a higher data rate.

The energy per operation is obtained by dividing the average power dissipation by the maximum input data rate at which each individual design is capable to operate. This index is useful for comparing the power consumptions of two designs working at different frequencies. For this purpose, the power consumption of each circuit operating on randomly generated input data at its minimum clock period is simulated. The energy per operation for each circuit can then be assessed by the product of this power and its critical path delay [Ali02]. The power-delay products of the proposed design, [Ye08], [Mey03] and [Ma10] for DR = 15, 18, 21 and 24 plotted in Figure 3.5 indicate that the proposed architecture is substantially more energy efficient than [Ye08], [Mey03] and [Ma10].

Table 3.7 Comparison of Synthesized Areas and Delays (Value in Parenthesis Indicates the Percentage Area or Delay Reduction of Proposed Design Over Its Contender)

DR (bit)	Synthesized Area (μm^2)				Synthesized Delay (ns)			
	This	[Ye08]	[Mey03]	[Ma10]	This	[Ye08]	[Mey03]	[Ma10]
15	9,094	11,599 (21.59)	71,977 (87.36)	87,152 (89.57)	2.00	2.81 (28.83)	12.92 (84.52)	5.42 (63.10)
18	9,164	17,643 (48.06)	151,378 (93.95)	278,000 (96.70)	1.94	3.02 (35.76)	18.87 (89.72)	6.80 (71.47)
21	10,096	20,710 (51.25)	242,571 (95.84)	648,937 (98.44)	2.09	3.26 (35.89)	25.50 (91.80)	8.30 (74.82)
24	13,725	21,652 (36.61)	427,602 (96.79)	2,143,865 (99.36)	2.00	3.34 (40.12)	31.83 (93.72)	9.10 (78.02)

Table 3.8 Comparison of Leakage and Total Power Dissipations (Value in Parenthesis Indicates the Percentage Leakage or Total Power Reduction of Proposed Design Over Its Contender)

DR (bit)	Leakage Power (nW)				Total Power (μ W)			
	This	[Ye08]	[Mey03]	[Ma10]	This	[Ye08]	[Mey03]	[Ma10]
15	454	585 (22.39)	2,899 (84.34)	3,013 (84.93)	673	996 (32.43)	26,790 (97.49)	6,431 (89.54)
18	449	806 (44.29)	5,490 (91.82)	8,506 (94.72)	529	1085 (51.24)	54,860 (99.04)	8,973 (94.10)
21	477	972 (50.93)	8,600 (94.45)	18,740 (97.45)	422	1059 (60.15)	61,330 (99.31)	10,870 (96.12)
24	681	1,051 (35.20)	1,402 (51.43)	59,710 (98.86)	440	908 (51.54)	80,580 (99.45)	19,730 (97.77)

Figure 3.5 Power-Delay Product for DR = 15, 18, 21 and 24 bits.

3.3.3 Physical Synthesis Results

Based on the prelayout synthesis results obtained from Table 3.6 and Table 3.7, the most competitive contender [Ye08] and the proposed design for the moduli set {255, 256, 257} were physically placed and routed using Cadence SoC Encounter ver. 8.1 to account for the parasitics of interconnections. As only the cores are to be compared, I/O pads were not inserted into the layouts of both designs. For a fair comparison, the timing constraint for each design was set to its respective minimum delay attained in Table 3.7, the initial core utilization ratios of both designs were set to be the same with four metal layers used for routing. After placement and

routing, the layouts were DRC verified and the extracted netlists were back-annotated for the same timing and power simulations at 1.62V and 3.34 ns clock period based on the Monte Carlo method [Bur93].

Both designs were able to achieve timing closure after the placement and routing. The physical synthesis results are summarized in Table 3.9. Based on the data extracted from Cadence SoC Encounter, the final core utilization ratios of the proposed design and [Ye08] are 0.636 and 0.685, respectively and the core size of the proposed design is 36.78% smaller than that of [Ye08], which is very close to the savings obtained from the pre-layout simulation. The denser interconnects of [Ye08] can be discerned from the layouts of the proposed design and [Ye08] shown in Figure 3.6 and Figure 3.7, respectively. Furthermore, the post-layout power dissipations show that the proposed design consumes 55.82% lower power than [Ye08]. The saving is slightly more than that obtained from the pre-layout simulation, which could be attributed to its lower interconnect complexity.

Table 3.9 Comparison of Post-layout Simulation Results

	This	[Ye08]
Core area (μm^2)	21050	33298
Delay (ns)	2.00	3.34
Leakage power (μW)	0.623	1.054
Total power (mW)	4.895	11.201
Final core utilization ratio	0.636	0.685

Figure 3.6 Layout of the proposed design.

Figure 3.7 Layout of the design of [Ye08].

3.4 Summary

Dynamic range reduction in different parts of the datapath has been used beneficially to boost the performance and reduce the power consumption of large system. The basic operations required for dynamic range reduction are sign detection and truncation, which can be trivially

performed in two's complement system but are compute intensive in RNS. In this chapter, a low complexity, high-speed and low-power 2^n signed integer RNS scaler for moduli set $\{2^n-1, 2^n, 2^n+1\}$ has been proposed. By simplifying and merging the expensive sign detection and correction circuits, the complexity of implementing the signed integer RNS scaler has been reduced substantially. The proposed design is benchmarked against several latest published signed integer RNS scalars [Ye08], [Mey03] and [Ma10] using the unit-gate analysis and logic synthesized results, and the most competitive contender was found to be [Ye08]. The logic synthesized results show that the proposed design is on average 39.38% smaller, 35.15% faster and 48.84% more power-efficient than [Ye08] over four different dynamic ranges. This superiority is further corroborated by the physical synthesis results based on the same TSMC $0.18 \mu\text{m}$ standard cell implementation for $n = 8$, where the proposed design runs 40% faster, consumes 56% lower power and occupies 37% lesser silicon area than [Ye08].

In the next chapter, a new concept of base transformation via injective mapping of residues in multi-base RNS system will be introduced and a new algorithm for mapping the residue representation of a signed integer to a different moduli set with smaller dynamic range will be proposed.

Chapter 4

Base Transformation with Injective Residue Mapping for Dynamic Range Reduction in RNS

4.1 Introduction

With Systems-on-Chip (SoC) integration, the impact of DSP core on die size has greatly reduced. To leverage the advantages of rescaling, more than one DSP cores with datasets of distinct dynamic ranges are integrated. One example is the integrated audio and video processing in a multi-media SoC. While audio is usually sampled at a low rate with high accuracy, video may have higher data flow and greater tolerance for processing errors. Their integration calls for arithmetic units of multiple precisions to optimize the overall processing cost, performance and output quality. The dynamic range of fixed point operations may also vary with the operational contexts. For example, a radar system used for navigation and guidance may require a wider dynamic range for tracking but need only a small subset of range for target acquisition and identification. With dynamic range fixed by a single moduli set, the advantages of RNS may not be fully exploited in these applications.

In RNS, the moduli set is selected upfront to cater for the dynamic range of the largest intermediate operations, which are usually exceedingly higher than the final output precision of the system. The change in dynamic range of intermediate computations of an application can usually be detected at design time by adopting some self-validated numeric analysis such as affine arithmetic [Sto03]. Since certain difficult inter modulus channel operations, such as sign detection and scaling, can be more efficiently carried out in the smaller than larger moduli set, base transformation from larger to smaller moduli set plays a pivotal role in the multi-base RNS. In [Jen93], it was proposed to vary the number of modulus channels when the dynamic ranges of intermediate computations change. Although this method incurs minimal hardware overhead on the additional buffers, it can only be implemented on serial-by-modulus residue

arithmetic operation, which is significantly slower. By removing one or more modulus channels to reduce the dynamic range, the sizes of residue arithmetic operators in the remaining modulus channels are not reduced, and the target dynamic range is severely limited by the product of the remaining moduli. For the special three-moduli set $\{2^n-1, 2^{n+k}, 2^n+1\}$, the delay of the residue-to-binary converter is comparable to that of the modulo multiplier in the critical modulus channel [Cho09]. As the transcoding is usually performed once at the end of the datapath, after applying several residue arithmetic operations, better performance can be achieved by using smaller arithmetic operators, while preserving parallelism [Cao10],[Sha07]. To have higher flexibility and gain in performance, the reduction in dynamic range should be accompanied by a reduction in the sizes of the moduli, which motivates the research for a more efficient base transformation than the downright approach of the reverse conversion from the large moduli set followed by a forward conversion to the small moduli set.

In this chapter, the word lengths of the residue operations in application-specific DSP are proposed to be reduced by base transformation. Using base transformation to prune the unused dynamic range enables the area, power and delay for all arithmetic operations to be simultaneously reduced in the new base, without sacrificing the parallelism and modularity of the original RNS base. This new paradigm of range adaptation is made possible for the first time with the proposed algorithm to efficiently remap the residues of a signed integer from an arithmetic friendly three-moduli set $S_1 \equiv \{2^n-1, 2^{n+k}, 2^n+1\}$ to another arithmetic friendly three-moduli set $S_2 \equiv \{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1\}$, where α is an integer in the range $[1, n-2]$. Since α is a positive integer, the dynamic range and the word length of each modulus of S_2 are always smaller than those of S_1 . As the parameters n , k and α can be changed to suit the desire dynamic range, S_1 and S_2 represent not just one but a rich combination of multi-base RNS possibilities. The proposed subranging algorithm directly maps the quantized segment of the residues in the domain of S_1 to its corresponding quantized segment in the codomain of S_2 , thus avoiding the need for a complex full reverse conversion and greatly simplify its sign detection. It results in significant saving in hardware resources compared with the best possible improvised solution, especially when the dynamic range reduction is large, i.e., when α is high.

This chapter is organized as follows. Section 4.2 introduces the concept of injective residue mapping for base transformation. The proposed algorithm and the architecture for its implementation are presented in Section 4.3. In Section 4.4, the proposed design is evaluated against the best improvised solution based on unit-gate model and synthesis results. Section 4.5 summarizes the chapter and discusses possible expansion of the proposed algorithm.

4.2 Injective Mappings for Base Transformation

Let $(x_1, x_2, \dots, x_N)$ be the residue representation of an unsigned integer X in an RNS of base $\{m_1, m_2, \dots, m_N\}$ with dynamic range $M = \prod_{i=1}^N m_i$, and $(x'_1, x'_2, \dots, x'_K)$ be the RNS representation of X under a different base $\{m'_1, m'_2, \dots, m'_K\}$ with a dynamic range $M' = \prod_{i=1}^K m'_i$, where N and K can be any positive integer larger than 1. Figure 4.1(a) shows the domain and codomain of mapping an unsigned integer X from the RNS of base $\{m_1, m_2, \dots, m_N\}$ to the RNS of base $\{m'_1, m'_2, \dots, m'_K\}$ for $M > M'$. The most intuitive way of accomplishing this is to perform a reverse conversion from $(x_1, x_2, \dots, x_N)$ to X , followed by a forward conversion of $|X|_{m'_i}$ to obtain $(x'_1, x'_2, \dots, x'_K)$. For an injective mapping, X has to be bounded by M' . This criterion can be assured by scaling X before the residue mapping if it falls outside the bound M' .

Figure 4.1 Base transformation of (a) unsigned integer X ; and (b) signed integer $\tilde{X}$ for $M > M'$.

Remapping of the residue representation of signed integer $\tilde{X}$ from the base $\{m_1, m_2, \dots, m_N\}$ to the base $\{m'_1, m'_2, \dots, m'_K\}$ is more involved. It requires additional computations in comparison with the unsigned counterpart. Figure 4.1(b) shows the domain and codomain of the mapping of $\tilde{X}$ from an RNS base with a larger dynamic range into one with a smaller dynamic range. The injection in the residue domain can be performed by first computing the magnitude X and the sign of $\tilde{X}$ from $(x_1, x_2, \dots, x_N)$ before computing $\tilde{X}$ by Equation (2.4) (see Section 2.1.1 of Chapter 2). The value of X recovered from the residue representation of $\tilde{X}$ is different for different bases. $\tilde{X}$ in $\{m_1, m_2, \dots, m_N\}$ is related to X' in $\{m'_1, m'_2, \dots, m'_K\}$ by:

$$\tilde{X} = X' - s \cdot M' \quad (4.1)$$

Substituting Equation (4.1) into Equation (2.4) results in:

$$\begin{aligned} X - s \cdot M &= X' - s \cdot M' \\ X' &= X - s \cdot (M - M') \end{aligned} \quad (4.2)$$

After the computation of X' from X by Equation (4.2), each residue of $(x'_1, x'_2, \dots, x'_K)$ can be generated by $|X'|_{m'_i}$. For symmetric signed residue representation, the positive and negative integers will fall into the lower and upper half ranges, as shown in Figure 4.2 for the mapping of the residues of X to those of X' . The mapping is a bijection for all $X' \in [0, M' - 1]$.

Figure 4.2 Mapping of $(x_1, x_2, \dots, x_N)$ to $(x'_1, x'_2, \dots, x'_K)$ for $M > M'$.

4.3 Proposed Base Transformation Algorithm and Architecture for Dynamic Range Reduction

The proposed algorithm aims to transform the base of the residues of a signed integer,

$$\tilde{X} \in \left[-\frac{M'}{2}, \frac{M'}{2} - 1 \right], \text{ from } S_1 \equiv \{2^n - 1, 2^{n+k}, 2^{n+1}\} \text{ to } S_2 \equiv \{2^{n-\alpha} - 1, 2^{n+k-\alpha}, 2^{n-\alpha} + 1\}, \text{ where } k \in [0,$$

$n]$ and $\alpha \in [1, n-2]$. It should be noted that with different combinations of k and α , S_1 and S_2

represent a large number of modular arithmetic friendly three-moduli sets. Since α is a positive

integer greater than or equal to 1, the dynamic range $M = 2^{n+k} (2^{2n} - 1)$ of S_1 is always greater

than the dynamic range $M' = 2^{n+k-\alpha} (2^{2(n-\alpha)} - 1)$ of S_2 . For $\tilde{X}$ to have a unique residue

representation in S_2 , it is assumed to be an integer within the range $\left[-\frac{M'}{2}, \frac{M'}{2} - 1 \right]$.

4.3.1 Algorithm

Let (x'_1, x'_2, x'_3) be the residue representation of $\tilde{X}$ in S_2 . The residues can be computed from X

by modulo reduction of X' using Equation (4.2):

$$x'_1 = |X'|_{2^{n-\alpha}-1} = |X - s \cdot M|_{2^{n-\alpha}-1} \quad (4.3)$$

$$x'_2 = |X'|_{2^{n+k-\alpha}} = |X|_{2^{n+k-\alpha}} = |x_2|_{2^{n+k-\alpha}} \quad (4.4)$$

$$x'_3 = |X'|_{2^{n-\alpha}+1} = |X - s \cdot M|_{2^{n-\alpha}+1} \quad (4.5)$$

From Equation (4.4), the binary representation of x'_2 corresponds to the least significant

$n+k-\alpha$ bits of x_2 . According to Equations (4.3) and (4.5), x'_1 and x'_3 cannot be computed

directly from the residues (x_1, x_2, x_3) of $\tilde{X}$ in S_1 without having to compute X first.

To avoid a full residue-to-binary conversion for X , the range detection method [Tay14] is used

to determine s and eliminate many unnecessary computations when α is large. Let X_{13} be the

magnitude corresponding to the residues (x_1, x_3) in a two-moduli set RNS $\{m_1, m_3\}$. By

treating $\{m_1 \times m_3, m_2\}$ as a two-moduli set, X can be computed from (X_{13}, x_2) by Equation (2.12)

(See Section 2.1.3 of Chapter 2):

$$\begin{aligned}
X &= X_{13} + (m_1 \times m_3) \left\| (m_1 \times m_3)^{-1} \right\|_{m_2} (x_2 - X_{13}) \Big|_{m_2} \\
&= X_{13} + (2^{2n} - 1) \left\| (2^{2n} - 1)^{-1} \right\|_{2^{n+k}} (x_2 - X_{13}) \Big|_{2^{n+k}} \\
&= X_{13} + (2^{2n} - 1) (-1) (x_2 - X_{13}) \Big|_{2^{n+k}} \\
&= X_{13} + (2^{2n} - 1) \beta
\end{aligned} \tag{4.6}$$

where the multiplicative inverse of $m_1 \times m_3$ is $\left\| (2^{2n} - 1)^{-1} \right\|_{2^{n+k}} = -1$ and $\beta = |X_{13} - x_2|_{2^{n+k}}$.

Since $M = 2^{n+k} (2^{2n} - 1)$, M can be quantized into 2^{n+k} divisions [Tay14], each with an equal range of $2^{2n} - 1$ integers as shown in Figure 4.3. From Equation (4.6), each division can be uniquely tagged by $\beta \in [0, 2^{n+k} - 1]$. It should be noted that not all divisions tagged by β are occupied by X due to the criterion for injective mapping of $\tilde{X}$ from S_1 to S_2 .

Figure 4.3 Division of M into 2^{n+k} subranges of $2^{2n}-1$ integers.

As M for the considered moduli set S_1 is always even, if $\tilde{X}$ is negative, $X \geq M/2 = 2^{n+k-1} (2^{2n} - 1)$ based on the condition for $s = 1$ in Equation (2.5) (See Section 2.1.1 of Chapter 2). According to Equation (4.6), X is greater than or equal to $M/2$ when $\beta \geq 2^{n+k-1}$. In other words, s can be determined from the MSB of β , i.e., $s = \beta_{n+k-1}$ (the index, $n+k-1$, represents the position of the binary bit of β).

It is further noted that the domain of X is confined to $X \in [0, M'/2] \cup [M - M'/2, M]$ for injective mapping, and the middle divisions will not be occupied. Each half range of the codomain M' is mapped from an equal number of valid divisions at the two tails of the

domain M , as shown in Figure 4.2. Since the size of each division is $2^{2n} - 1$, the number of valid divisions r can be calculated as follows:

$$\begin{aligned}
r &= 2 \left\lceil \frac{M'/2}{2^{2n} - 1} \right\rceil = 2 \left\lceil \frac{2^{n+k-\alpha-1} (2^{2n-2\alpha} - 1)}{2^{2n} - 1} \right\rceil \\
&= 2 \left\lceil \frac{2^{n+k-\alpha-1} [2^{-2\alpha} (2^{2n} - 1) - 1 + 2^{-2\alpha}]}{2^{2n} - 1} \right\rceil \\
&= 2 \left\lceil 2^{n+k-3\alpha-1} - 2^{n+k-\alpha-1} \left(\frac{1 - 2^{-2\alpha}}{2^{2n} - 1} \right) \right\rceil
\end{aligned} \tag{4.7}$$

where $\lceil y \rceil$ denotes the smallest integer greater than or equal to y .

As $k \leq n$ and $\alpha \geq 1$, the second term of Equation (4.7) is always less than 1. Thus, Equation (4.7) can be simplified to

$$\begin{aligned}
r &= 2 \lceil 2^{n+k-3\alpha-1} \rceil \\
r &= \begin{cases} 2^{n+k-3\alpha} & \text{if } n+k-3\alpha > 0 \\ 2 & \text{if } n+k-3\alpha \leq 0 \end{cases}
\end{aligned} \tag{4.8}$$

Equation (4.8) shows that r is dependent on α , which is a parameter of the transformed base. A larger α means a smaller dynamic range M' of the target moduli set S_2 , and hence a smaller r .

Due to the symmetry of the signed residue representations, it turns out that the sign s of $\tilde{X}$ and the exact divisions in which X falls can be readily identified by the least significant $\log_2 r$ bits of β . The remaining bits of β can be construed as its sign-extended bits. That is, β can be formed by the concatenation of two binary substrings:

$$\beta = \beta_H \parallel \beta_L \tag{4.9}$$

where β_L is the least significant $\log_2 r$ bits of β and β_H is a constant binary string with all bits equal to s . β_H is 3α bits long if $n+k-3\alpha > 0$, and $n+k-1$ bits long otherwise. In fact, s can be determined by the MSB of β_L , i.e., $s = \beta_{n+k-3\alpha-1}$.

Since $\beta_H = s \cdot (2^{n+k-\log_2 r} - 1) = s \cdot (2^{n+k}/r - 1)$ and $\beta_L = |\beta|_r$, β can be expressed as

$$\beta = \beta_H \times r + \beta_L = s \cdot (2^{n+k} - r) + |\beta|_r \quad (4.10)$$

Substituting Equation (4.10) into Equation (4.6),

$$\begin{aligned} X &= X_{13} + (2^{2n} - 1) \left[s \cdot (2^{n+k} - r) + |\beta|_r \right] \\ &= X_{13} + (2^{2n} - 1) |\beta|_r + s \cdot \left[M - r(2^{2n} - 1) \right] \end{aligned} \quad (4.11)$$

Comparing with the computation of X by Equation (4.6), only $\log_2 r$ bits of $\beta_L = |\beta|_r$, as opposed to $n+k$ bits of β , need to be computed. The sign bit s can be directly obtained from the MSB of β_L to gate the constant integer $M - r(2^{2n} - 1)$, which can be pre-computed once the moduli sets S_1 and S_2 have been selected.

By substituting Equation (4.11) into Equations (4.3) and (4.5), x'_1 and x'_3 can be calculated by:

$$\begin{aligned} x'_1 &= \left| X_{13} + (2^{2n} - 1) |\beta|_r - s \cdot r \cdot (2^{2n} - 1) \right|_{2^{n-\alpha}-1} \\ &= \left| X_{13} \right|_{2^{n-\alpha}-1} + 2^{2\alpha} |\beta|_r - |\beta|_r - C_1 \Big|_{2^{n-\alpha}-1} \end{aligned} \quad (4.12)$$

$$\begin{aligned} x'_3 &= \left| X_{13} + (2^{2n} - 1) |\beta|_r - s \cdot r \cdot (2^{2n} - 1) \right|_{2^{n-\alpha}+1} \\ &= \left| X_{13} \right|_{2^{n-\alpha}+1} + 2^{2\alpha} |\beta|_r - |\beta|_r - C_2 \Big|_{2^{n-\alpha}+1} \end{aligned} \quad (4.13)$$

where $C_1 = \left| s \cdot r (2^{2\alpha} - 1) \right|_{2^{n-\alpha}-1}$ and $C_2 = \left| s \cdot r (2^{2\alpha} - 1) \right|_{2^{n-\alpha}+1}$.

The following example illustrates the computation steps of the proposed algorithm.

Example 4.1: Assume $n = 5, k = 0, \alpha = 1, S_1 \equiv \{31, 32, 33\}$ and $S_2 \equiv \{15, 16, 17\}$. Since $n + k - \alpha > 0$, from Equation (4.8), r can be computed as $2^{5+0-3} = 4$. As shown in Figure 4.4, two divisions from each end of the range M are mapped respectively to the lower and upper half ranges of M' . Let the signed integer $\tilde{X} = -1987$. Its residue representation is $(x_1, x_2, x_3) \equiv (28, 29, 26)$ in S_1 . From Equation (4.4), x'_2 can be computed as $|29|_{16} = 13$. Since $X_{13} \equiv (28, 26)$ in RNS $\{31 \times 33, 32\}$, its magnitude is decoded to be 59. From the definition of β in Equation (4.6), $\beta = |59 - 29|_{32} = 30 \equiv 11110_2$. From Equation (4.9), $\beta_H = 111_2$ and $\beta_L = 10_2$. From the MSB of β_L , $s = 1$. Substituting X_{13} , β , and s into Equations (4.12) and (4.13),

$x'_1 = \left\| 59|_{15} + 2^2 |30|_4 - |30|_4 - |1 \times 4(2^2 - 1)|_{15} \right\|_{15} = 8$ and $x'_3 = \left\| 59|_{17} + 2^2 |30|_4 - |30|_4 - |1 \times 4(2^2 - 1)|_{17} \right\|_{17} = 2$. Thus, the proposed base transformation yields the residue representation of (8, 13, 2) in S_2 . It can be verified by reverse conversion that it corresponds to the integer -1987.

Figure 4.4 Mapping of valid divisions for base transformation from S_1 to S_2 for $n = 5$, $k = 0$ and $\alpha = 1$.

The above example shows that X_{13} , β_L and s need to be computed from (x_1, x_2, x_3) to obtain (x'_1, x'_2, x'_3) . Their hardware implementations will be discussed below.

4.3.2 Architecture

Figure 4.5 shows the architecture for the proposed base transformation algorithm. It consists of two major blocks for: (i) computing X_{13} and β_L , and (ii) remapping of residues. The former block consists of n modified full adders (MFAs), and two parallel prefix adders with a common fast carry input (FCI). The latter block is composed of an EAC CSA tree, an inverted EAC (IEAC) CSA tree, a mod $2^{n-\alpha} - 1$ adder, and a $(n-\alpha)$ -bit parallel prefix adder with FCI. The numbers of input operands to the EAC CSA and IEAC CSA trees are represented in Figure 4.5 by T_1 and T_2 , respectively. The derivation and implementations of these blocks are detailed as follows.

Figure 4.5 Architecture of the proposed base transformation algorithm.

$$\begin{aligned}
X_{13} &= \left| m_3 \left| m_3^{-1} \right|_{m_1} x_1 + m_1 \left| m_1^{-1} \right|_{m_3} x_3 \right|_{m_1 m_3} \\
&= \left| (2^n + 1)(2^{n-1})x_1 + (2^n - 1)(2^{n-1})x_3 \right|_{2^{2n}-1} \\
&= \left| (2^{2n-1} + 2^{n-1})x_1 + (2^{2n-1} - 2^{n-1})x_3 \right|_{2^{2n}-1} \\
&= \left| P_1 + P_2 \right|_{2^{2n}-1}
\end{aligned} \tag{4.14}$$

where $\left| m_3^{-1} \right|_{m_1} = 2^{n-1}$ is the multiplicative inverse of m_3 with respect to m_1 , $\left| m_1^{-1} \right|_{m_3} = 2^{n-1}$ is the

multiplicative inverse of m_1 with respect to m_3 , $P_1 = \left| 2^{2n-1}x_1 + 2^{n-1}x_1 \right|_{2^{2n}-1}$, and

$$P_2 = \left| 2^{2n-1}x_3 - 2^{n-1}x_3 \right|_{2^{2n}-1}.$$

By applying the circular left shift *Properties 2.5* and *2.6*, and merging zero substrings, P_1 and P_2 can be expressed as:

$$\begin{aligned}
P_1 &= \left| CLS_{2n}(x_1, 2n-1) + CLS_{2n}(x_1, n-1) \right|_{2^{2n-1}} \\
&= \left| x_{1,0} \underbrace{0 \cdots 0}_n \underbrace{x_{1,n-1} \cdots x_{1,1}}_{n-1} + 0 \underbrace{x_{1,n-1} \cdots x_{1,0}}_n \underbrace{0 \cdots 0}_{n-1} \right|_{2^{2n-1}} \\
&= x_{1,0} \underbrace{x_{1,n-1} \cdots x_{1,0}}_n \underbrace{x_{1,n-1} \cdots x_{1,1}}_{n-1}
\end{aligned} \tag{4.15}$$

$$\begin{aligned}
P_2 &= \left| CLS_{2n}(x_3, 2n-1) + CLS_{2n}(\bar{x}_3, n-1) \right|_{2^{2n-1}} \\
&= \left| x_{3,0} \underbrace{0 \cdots 0}_{n-1} \underbrace{x_{3,n} \cdots x_{3,1}}_n + \bar{x}_{3,n} \cdots \bar{x}_{3,0} \underbrace{1 \cdots 1}_{n-1} \right|_{2^{2n-1}}
\end{aligned} \tag{4.16}$$

From Equation (4.15), P_1 can be computed by simply rearranging the bits of x_1 . On the other hand, the expression of P_2 in Equation (4.16) is dependent on $x_{3,n}$. Since $x_3 \in [0, 2^n]$, when $x_3 = 2^n$, $x_{3,n} = '1'$ and $x_{3,i} = '0' \forall i \leq n-1$. Under this condition, P_2 can be expressed as a constant C' as shown below.

$$\begin{aligned}
P_2 &= \left| \underbrace{0 \cdots 0}_n \underbrace{10 \cdots 0}_{n-1} + \underbrace{01 \cdots 11}_{n-1} \underbrace{1 \cdots 1}_{n-1} \right|_{2^{2n-1}} \\
&= \underbrace{10 \cdots 01}_n \underbrace{1 \cdots 1}_{n-1} = C'
\end{aligned} \tag{4.17}$$

where $C' = \underbrace{10 \cdots 01}_n \underbrace{1 \cdots 1}_{n-1}$. When $x_3 < 2^n$, $x_{3,n} = '0'$. Therefore,

$$\begin{aligned}
P_2 &= \left| x_{3,0} \underbrace{0 \cdots 0}_n \underbrace{x_{3,n-1} \cdots x_{3,1}}_{n-1} + 1 \underbrace{\bar{x}_{3,n-1} \cdots \bar{x}_{3,0}}_n \underbrace{1 \cdots 1}_{n-1} \right|_{2^{2n-1}} \\
&= \left| x_{3,0} \underbrace{\bar{x}_{3,n-1} \cdots \bar{x}_{3,0}}_n \underbrace{x_{3,n-1} \cdots x_{3,1}}_{n-1} + \underbrace{10 \cdots 01}_n \underbrace{1 \cdots 1}_{n-1} \right|_{2^{2n-1}} \\
&= |P'_2 + C'|_{2^{2n-1}}
\end{aligned} \tag{4.18}$$

where $P'_2 = x_{3,0} \underbrace{\bar{x}_{3,n-1} \cdots \bar{x}_{3,0}}_n \underbrace{x_{3,n-1} \cdots x_{3,1}}_{n-1}$.

Substituting Equations (4.15), (4.17) and (4.18) into Equation (4.14), X_{13} can be computed by:

$$X_{13} = \begin{cases} |P_1 + C'|_{2^{2n-1}} & \text{if } x_{3,n} = '1' \\ |P_1 + P'_2 + C'|_{2^{2n-1}} & \text{if } x_{3,n} = '0' \end{cases} \tag{4.19}$$

The mod $2^{2n} - 1$ addition of Equation (4.19) is illustrated in Figure 4.6. It consists of two stages. Stage 1 performs a simplified carry save addition of two variables and one constant $2n$ -bit vectors. The addition operations at each bit position are divided into two groups, $x_{1,i} + x_{3,i} + 1$ and $x_{1,i} + (\bar{x}_{3,i} \cdot \bar{x}_{3,n}) \forall i \in [0, n-1]$, which can be performed by the MFA shown in Figure 4.7. The MFA is made up of two modified half adders (MHAs), denoted by MHA1 and MHA2, which are used to compute the sum and carry bits for the two groups. The sum and carry bits are then rearranged using the bit rewiring module in Figure 4.5 to form the $2n$ -bit sum and carry vectors, A and B , for the subsequent addition operation in Stage 2 of Figure 4.6. Due to the end around carry of mod $2^{2n} - 1$ addition in Stage 1, the vacant LSB position of the carry vector B is replaced by its MSB bit, B_{2n} , in Stage 2, as indicated by the arrow in Figure 4.6. Thus, the mod $2^{2n} - 1$ addition in Stage 2 can be written as:

$$\begin{aligned} X_{13} &= |A + B|_{2^{2n}-1} \\ &= |A + B + c_o|_{2^{2n}} \end{aligned} \quad (4.20)$$

where $c_o \in \{0, 1\}$ is the end around carry, which is the carry output of the binary addition of A and B .

Equation (4.20) can be implemented using a $2n$ -bit parallel prefix adder with c_o as the FCI.

Figure 4.6 Addition stages required for X_{13} computation.

Figure 4.7 Bit slices of MHA1 and MHA2 in Figure 4.6 and MFA in Figure 4.5.

Since $\beta = |X_{13} - x_2|_{2^{n+k}}$ (see its definition in Equation (4.6)), β_L can be computed by:

$$\begin{aligned} \beta_L &= |\beta|_r = \left| |X_{13} - x_2|_{2^{n+k}} \right|_r = |A + B + c_o - x_2|_r \\ &= |A + B + (\bar{x}_2 + 1) + c_o|_r \end{aligned} \quad (4.21)$$

Figure 4.8 illustrates the modulo r addition of Equation (4.21). Since r is a power-of-two integer, modulo r addition is equivalent to a binary addition with the carry output discarded. Stage 1 performs the carry save addition of the first three terms of Equation (4.21). The resulting sum and carry vectors are denoted by D and E , respectively. Stage 2 computes $|D + E + 1 + c_o|_r$. It can be implemented using a $(\log_2 r)$ -bit parallel prefix adder with c_o as the FCI. If $n + k - 3\alpha \leq 0$, the circuit can be simplified to only three XOR gates.

	A_{b-1}	A_{b-2}	$\cdots$	A_1	A_0	
	B_{b-1}	B_{b-2}	$\cdots$	B_1	B_0	
	$\bar{x}_{2,b-1}$	$\bar{x}_{2,b-2}$	$\cdots$	$\bar{x}_{2,1}$	$\bar{x}_{2,0}$	Stage 1
+	<hr/>					
	D_{b-1}	D_{b-2}	$\cdots$	D_1	D_0	
	E_{b-1}	E_{b-2}	$\cdots$	E_1	1	Stage 2
+	<hr/>					
	c_o					

Figure 4.8 Addition stages for the computation of β_L .

Figure 4.9 shows the complete architecture of the parallel prefix adders with FCI used for the computations of X_{13} and β_L for $n = 4$, $k = 2$ and $\alpha = 1$. The carry output signal c_o is shared by both prefix adders.

Figure 4.9 Parallel prefix adders with FCI for $n = 4$, $k = 2$ and $\alpha = 1$.

The block ‘remapping of residues’ in Figure 4.5 computes (x'_1, x'_2, x'_3) from x_2 , X_{13} , β_L and s . According to (4.4), x'_2 can be obtained from the input x_2 directly without requiring any hardware. The remaining two residues, x'_1 and x'_3 , in base S_2 are computed from X_{13} , β_L , and s according to Equations (4.12) and (4.13). The role of the bit rewiring modules in the m'_1 and m'_3 modulus channels of the remapping of residues block is to rearrange the bits of X_{13} , β_L and s before they are fed into EAC CSA and IEAC CSA trees for subsequent additions, while the bit rewiring module in the m'_2 modulus channel simply routes the least significant $n + k - \alpha$ bits of x_2 to x'_2 . To facilitate the modulo $2^{n-\alpha} - 1$ reduction of X_{13} , the $2n$ -bit integer X_{13} is decomposed into a weighted sum of $(n-\alpha)$ -bit integers. Let $X_{13}^{(i)}$ denote the i -th substring of $n-\alpha$ bits. Then $X_{13} = X_{13}^{(l)} \parallel \dots \parallel X_{13}^{(1)} \parallel X_{13}^{(0)}$ can be expressed as:

$$X_{13} = 2^{l \times (n-\alpha)} X_{13}^{(l)} + \dots + 2^{n-\alpha} X_{13}^{(1)} + 2^0 X_{13}^{(0)} \quad (4.22)$$

where $l = \lceil 2n / (n - \alpha) \rceil - 1$.

Substituting Equation (4.22) into Equation (4.12), since $\left| 2^{j(n-\alpha)} \right|_{2^{n-\alpha}-1} = 1 \quad \forall j \geq 0$,

$$x'_1 = \left| X_{13}^{(l)} + \dots + X_{13}^{(1)} + X_{13}^{(0)} + 2^{2\alpha} \beta_L - \beta_L - C_1 \right|_{2^{n-\alpha}-1} \quad (4.23)$$

The word length of β_L is $\log_2 r$, which may exceed $n - \alpha$ bits. To ease the modulo $2^{n-\alpha} - 1$ reduction, it can also be decomposed into a weighted sum of $j+1$ sub-integers where $j = \lceil \log_2 r / (n - \alpha) \rceil - 1$. From the expression of r in Equation (4.8) and $k \leq n$,

$$\max(j) = \max \left(\left\lceil \frac{n+k-3\alpha}{n-\alpha} \right\rceil - 1 \right) = \left\lceil \frac{n+n-3\alpha}{n-\alpha} \right\rceil - 1 = 1 \quad (4.24)$$

When $j = 1$, $\beta_L = \beta_L^{(1)} \parallel \beta_L^{(0)} = 2^{n-\alpha} \beta_L^{(1)} + \beta_L^{(0)}$. By applying *Properties 1* and *2*, Equation (4.23)

can then be simplified to:

$$\begin{aligned} x'_1 &= \left| X_{13}^{(l)} + \dots + X_{13}^{(1)} + X_{13}^{(0)} + 2^{2\alpha} (\beta_L^{(1)} + \beta_L^{(0)}) - \beta_L^{(1)} - \beta_L^{(0)} - C_1 \right|_{2^{n-\alpha}-1} \\ &= \left| X_{13}^{(l)} + \dots + X_{13}^{(1)} + X_{13}^{(0)} + CLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha) + CLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \overline{\beta_L^{(1)}} + \overline{\beta_L^{(0)}} + \overline{C_1} \right|_{2^{n-\alpha}-1} \end{aligned} \quad (4.25)$$

When $j = 0$, $\beta_L = \beta_L^{(0)}$. Hence,

$$x'_1 = \left| X_{13}^{(l)} + \dots + X_{13}^{(1)} + X_{13}^{(0)} + CLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \overline{\beta_L^{(0)}} + \overline{C_1} \right|_{2^{n-\alpha}-1} \quad (4.26)$$

These modular additions can be implemented by using $(n-\alpha)$ -bit EAC CSA tree followed by a mod $2^{n-\alpha} - 1$ adder as shown in Figure 4.5. From Equations (4.25) and (4.26), the number of input operands T_1 of the $(n-\alpha)$ -bit EAC CSA tree is given by:

$$T_1 = \begin{cases} l+6 & \text{if } j=1 \\ l+4 & \text{if } j=0 \end{cases} \quad (4.27)$$

Since $\left| 2^{j(n-\alpha)} \right|_{2^{n-\alpha}+1} = (-1)^j \forall j \geq 0$, x'_3 can be similarly computed by decomposing X_{13} and β_L into

weighted sums of $(n-\alpha)$ -bit integers to simplify the modulo reduction. By substituting Equation (4.22) and $\beta_L = 2^{n-\alpha} \beta_L^{(1)} + \beta_L^{(0)}$ into Equation (4.13),

$$\begin{aligned} x'_3 &= \left| 2^{l \times (n-\alpha)} X_{13}^{(l)} + \dots + 2^{n-\alpha} X_{13}^{(1)} + 2^0 X_{13}^{(0)} + 2^{2\alpha} (2^{n-\alpha} \beta_L^{(1)} + \beta_L^{(0)}) - 2^{n-\alpha} \beta_L^{(1)} - \beta_L^{(0)} - C_2 \right|_{2^{n-\alpha}+1} \\ &= \left| (-1)^l X_{13}^{(l)} + \dots - X_{13}^{(1)} + X_{13}^{(0)} + 2^{2\alpha} (-\beta_L^{(1)} + \beta_L^{(0)}) + \beta_L^{(1)} - \beta_L^{(0)} - C_2 \right|_{2^{n-\alpha}+1} \end{aligned} \quad (4.28)$$

By applying *Property 3* on Equation (4.28),

$$\begin{aligned}
 x'_3 &= \left\lfloor \begin{aligned} &(-1)^l X_{13}^{(l)} + \dots - X_{13}^{(1)} + X_{13}^{(0)} - CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha) + \\ &CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + 2^{2\alpha} - 1 - (2^{2\alpha} - 1) + \beta_L^{(1)} - \beta_L^{(0)} - C_2 \end{aligned} \right\rfloor_{2^{n-\alpha}+1} \\
 &= \left\lfloor \begin{aligned} &(-1)^l X_{13}^{(l)} + \dots - X_{13}^{(1)} + X_{13}^{(0)} - CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha) \\ &+ CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \beta_L^{(1)} - \beta_L^{(0)} - C_2 \end{aligned} \right\rfloor_{2^{n-\alpha}+1}
 \end{aligned} \quad (4.29)$$

When l is even, Equation (4.29) can be simplified by *Property 4* to

$$x'_3 = \left\lfloor \begin{aligned} &X_{13}^{(l)} + \dots + \overline{X_{13}^{(1)}} + X_{13}^{(0)} + \overline{CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha)} + \\ &CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \beta_L^{(1)} + \overline{\beta_L^{(0)}} + \overline{C_2} + (l+6) \end{aligned} \right\rfloor_{2^{n-\alpha}+1} \quad (4.30)$$

A constant correction factor of $l+6$ is introduced in Equation (4.30) due to the application of *Property 4*. In [Efs12], the multi-operand mod $2^n + 1$ adder is simplified by introducing an all zeros n -bit vector. Using this technique, the correction factor of $l+6$ in Equation (4.30) can be eliminated by introducing an additional all zeros $(n-\alpha)$ -bit vector to make the number of input operands equal to $l+7$, which can be reduced by an IEAC CSA tree of $l+5$ $(n-\alpha)$ -bit IEAC CSAs. Let A and B be the final sum and carry vectors generated by the IEAC CSA tree. As each $(n-\alpha)$ -bit IEAC CSA will introduce a correction factor of -1 , the following expression can be obtained.

$$\begin{aligned}
 &\left\lfloor A + B - (l+5) \right\rfloor_{2^{n-\alpha}+1} \\
 &= \left\lfloor \begin{aligned} &X_{13}^{(l)} + \dots + \overline{X_{13}^{(1)}} + X_{13}^{(0)} + \overline{CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha)} + \\ &+ CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \beta_L^{(1)} + \overline{\beta_L^{(0)}} + \overline{C_2} + 0 \end{aligned} \right\rfloor_{2^{n-\alpha}+1}
 \end{aligned} \quad (4.31)$$

When l is odd,

$$x'_3 = \left\lfloor \begin{aligned} &X_{13}^{(l)} + \dots + \overline{X_{13}^{(1)}} + X_{13}^{(0)} + \overline{CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha)} + \\ &CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \beta_L^{(1)} + \overline{\beta_L^{(0)}} + \overline{C_2} + (l+7) \end{aligned} \right\rfloor_{2^{n-\alpha}+1} \quad (4.32)$$

In this case, the number of operands of the IEAC CSA tree has to be $l+8$ in order to eliminate the correction factor of $l+7$. By applying the same technique, two all zeros $(n-\alpha)$ -bit vectors are

added. The final sum and carry vectors of the IEAC CSA tree, as well as the correction factor introduced by the $l+6$ $(n-\alpha)$ -bit IEAC CSAs is given by:

$$\left| A + B - (l+6) \right|_{2^{n-\alpha}+1} = \left| \overline{X_{13}^{(l)}} + \dots + \overline{X_{13}^{(1)}} + X_{13}^{(0)} + \overline{CCLS_{n-\alpha}(\beta_L^{(1)}, 2\alpha)} + \right. \\ \left. CCLS_{n-\alpha}(\beta_L^{(0)}, 2\alpha) + \beta_L^{(1)} + \overline{\beta_L^{(0)}} + \overline{C_2} + 0 + 0 \right|_{2^{n-\alpha}+1} \quad (4.33)$$

Combining both cases, the number of input operands T_2 of the $(n-\alpha)$ -bit IEAC CSA tree can be expressed as:

$$T_2 = \begin{cases} l+7 & \text{if } l \text{ is even} \\ l+8 & \text{if } l \text{ is odd} \end{cases} \quad (4.34)$$

By substituting Equation (4.31) into Equation (4.30) for even l , and Equation (4.33) into Equation (4.32) for odd l , the following equation is obtained.

$$x'_3 = \left| A + B + 1 \right|_{2^{n-\alpha}+1} \quad (4.35)$$

Thus, x'_3 can be obtained by using a mod $2^{n-\alpha}+1$ adder to add the sum and carry vectors with a carry input of 1. A better implementation of this final modular adder is realized by a $(n-\alpha)$ -bit parallel prefix adder with FCI, as indicated in Figure 4.5. Based on the analysis of [Efs12], this method of implementation not only reduces the hardware complexity of the final modular adder but also decreases the number of IEAC CSAs required by the IEAC CSA tree.

4.4 Evaluation and Comparison

To the best of the author's knowledge, this is the first attempt to directly remap the residues from one moduli set to another. In the absence of any prior art, the best possible improvisation is constructed from the most efficient residue-to-binary converter of moduli set S_1 , a comparator with constant to detect the sign s from the binary representation of X , and the most efficient residue generators for moduli set S_2 . The improvised architecture is shown in Figure 4.10.

Figure 4.10 Architecture in comparison.

The residue-to-binary converter is implemented by the architecture proposed in [Cha04b]. The residue x'_2 is directly wired from x_2 . The mod $2^{n-\alpha}-1$ residue generator is implemented by an EAC CSA tree and mod $2^{n-\alpha}-1$ adder. The latter is implemented by the design proposed in [Che09]. The mod $2^{n-\alpha}-1$ residue generator is implemented by the architecture proposed in [Efs12]. By changing the values of n , k , and α , many different combinations of moduli sets for S_1 and S_2 can be formed. For a dynamic range M of S_1 ranging from 24 to 80 bits, $n = 8, 12, 16$ and 20 are considered. For each value of n , three different values of k are considered: $k = 0, n/2$ and n . For the chosen n and k , to ensure that M' of S_2 is not too small relative to M , $\alpha = 1, n/4$ and $n/2$ are considered.

4.4.1 Unit-gate Analysis

The hardware area and delay of the proposed and reference designs are first estimated by the unit-gate model [Che11]. The unit-gate areas and delays of the proposed (This) and reference (Ref) designs are compared in Table 4.1, where $\% \delta = (\text{Ref} - \text{This}) / \text{Ref} \times 100\%$. In general, the proposed design has slightly larger unit-gate area than the reference design when $\alpha = 1$. When $\alpha > 1$, the proposed design is always smaller than the reference design, with up to 38.65% of improvement. Owing to the dominant area reduction of the merged X_{13} and β_L computation, the percentage area reduction increases with α except for a few outliers. In addition, the proposed design is always faster, with the improvement ranging from 21.05% to 32.88%. This is

attributed to the avoidance of full residue-to-binary conversion and simplified sign detection.

Table 4.1 Comparison of Unit-gate Area and Delay.

n	k	α	Unit-gate Area		% δ	Unit-gate Delay		% δ
			This	Ref		This	Ref	
8	0	1	1080.5	1088.5	0.73	43	59	27.12
	0	2	916	1005	8.86	43	59	27.12
	0	4	757	938	19.30	45	57	21.05
	4	1	1264	1130.5	-11.81	45	62	27.42
	4	2	1003	1089	7.90	45	62	27.42
	4	4	757	1008	24.90	41	60	31.67
	8	1	1361.5	1237.5	-10.02	47	62	24.19
	8	2	1180	1273	7.31	47	62	24.19
	8	4	821	1150	28.61	45	64	29.69
12	0	1	1785	1769.5	-0.88	47	66	28.79
	0	3	1444	1596.5	9.55	47	66	28.79
	0	6	1188	1487	20.11	49	64	23.44
	6	1	2078.5	1862.5	-11.60	49	68	27.94
	6	3	1570	1752.5	10.41	47	68	30.88
	6	6	1188	1622	26.76	45	66	31.82
	12	1	2236	2068.5	-8.10	51	69	26.09
	12	3	1835.5	2070.5	11.35	49	69	28.99
	12	6	1297	1877	30.90	49	71	30.99
16	0	1	2456.5	2421.5	-1.45	49	66	25.76
	0	4	1933	2162	10.59	47	66	28.79
	0	8	1583	2016	21.48	49	64	23.44
	8	1	2866	2585.5	-10.85	51	69	26.09
	8	4	2113	2410	12.32	49	69	28.99
	8	8	1583	2236	29.20	45	67	32.84
	16	1	3046	2918.5	-4.37	51	69	26.09
	16	4	2479	2890	14.22	51	69	26.09
	16	8	1731	2632	34.23	49	71	30.99
20	0	1	3231.5	3206.5	-0.78	51	72	29.17
	0	5	2477.5	2803.5	11.63	49	70	30.00
	0	10	2068	2651	21.99	53	70	24.29
	10	1	3748	3461.5	-8.28	53	73	27.40
	10	5	2702.5	3163.5	14.57	49	71	30.99
	10	10	2068	2976	30.51	49	71	30.99
	20	1	4028.5	3949.5	-2.00	55	75	26.67
	20	5	3160	3833.5	17.57	51	73	30.14
	20	10	2270	3541	35.89	53	77	31.17

4.4.2 Synthesis Results

Both designs were described in Verilog HDL and synthesized by Synopsys Design Compiler version E-2010.03 based on STM 65nm CMOS technology library (CORE65GPSVT, ver. 5.1). Recursive speed optimizations of each circuit are performed to obtain its minimum delay. The results are shown in Table 4.2.

Table 4.2 Comparison of Synthesized Area and Delay.

n	k	α	Area (μm^2)		% δ	Delay (ns)		% δ
			This	Ref		This	Ref	
8	0	1	4601.48	4655.04	1.15	1.45	1.54	5.84
	0	2	3957.72	4777.24	17.15	1.42	1.52	6.58
	0	4	3670.16	3642.60	-0.76	1.18	1.50	21.33
	4	1	4845.88	5194.28	6.71	1.61	1.53	-5.23
	4	2	4633.72	5014.88	7.60	1.45	1.57	7.64
	4	4	3732.56	4650.88	19.75	1.18	1.55	23.87
	8	1	6472.44	5131.88	-26.12	1.51	1.60	5.63
	8	2	5119.40	5730.92	10.67	1.55	1.57	1.27
	8	4	3459.04	4405.96	21.49	1.38	1.62	14.81
12	0	1	7177.56	8388.12	14.43	1.60	1.68	4.76
	0	3	5648.76	6910.28	18.26	1.62	1.72	5.81
	0	6	5458.96	6353.88	14.08	1.55	1.64	5.49
	6	1	8289.32	8453.12	1.94	1.70	1.71	0.58
	6	3	6010.16	6760.52	11.10	1.65	1.74	5.17
	6	6	5781.88	7034.56	17.81	1.53	1.67	8.38
	12	1	8872.76	8324.68	-6.58	1.72	1.75	1.71
	12	3	8166.08	8349.12	2.19	1.68	1.73	2.89
	12	6	6016.40	6321.64	4.83	1.55	1.76	11.93
16	0	1	9080.24	9964.24	8.87	1.77	1.78	0.56
	0	4	8396.44	8822.32	4.83	1.70	1.82	6.59
	0	8	7002.32	8160.06	14.19	1.45	1.75	17.14
	8	1	10631.92	11363.04	6.43	1.83	1.83	0.00
	8	4	8813.48	9421.36	6.45	1.74	1.85	5.95
	8	8	8320.00	9221.68	9.78	1.42	1.78	20.22
	16	1	12215.84	12002.64	-1.78	1.83	1.85	1.08
	16	4	9880.00	10361.52	4.65	1.88	1.88	0.00
	16	8	7529.08	10568.48	28.76	1.68	1.80	6.67
20	0	1	11279.32	12895.48	12.53	1.83	1.88	2.66
	0	5	9510.28	10697.44	11.10	1.82	1.88	3.19
	0	10	5924.36	10417.16	43.13	1.76	1.80	2.22
	10	1	12744.16	13239.72	3.74	1.93	1.86	-3.76
	10	5	10729.68	11735.36	8.57	1.83	2.02	9.41
	10	10	9079.20	11454.16	20.73	1.75	1.90	7.89
	20	1	16631.16	13163.80	-26.34	1.90	1.95	2.56
	20	5	14057.16	11930.36	-17.83	1.90	1.98	4.04
	20	10	9576.84	10784.80	11.20	1.76	1.95	9.74

The synthesized area shows similar trend of increasing the percentage of area reduction of the proposed design with α , as in the unit-gate analysis. Unlike the unit-gate analysis, the proposed design is not always faster. This is mainly due to the Ladner-Fisher parallel-prefix structure used for the implementation of some of the modular adders in the proposed design, which generally has larger fan-out when the operands are large. The percentage reductions of area-delay product are plotted in Figure 4.11. There are four sections in the chart corresponding to $n = 8, 12, 16$ and 20 . The integers marked on the x -axis of each section refer to the values of k for each n . The blue, green and red bars correspond to $\alpha = 1, n/4$ and $n/2$, respectively. It is evident that the reduction in area-delay product increases with α for all cases except for a few outliers, which implies that the area-time efficiency of the proposed design is more prominent when the base transformation results in larger dynamic range reduction.

Figure 4.11 Percentage reduction of area-delay product.

The power consumptions of both designs operating at their respective maximum frequency are also measured using Synopsys PrimeTime PX. Monte Carlo simulation [Pie10b] is used where a finite number of randomly generated inputs are fed into a design for estimating the power consumptions. In this experiment, 500 random input vectors were injected into both designs, with 99% confidence level that the error of average power estimation is less than 2.5%. The total and leakage power consumptions of both designs are compared in Table 4.3. Since the designs have different speeds, their energy consumptions per operation are also compared. The percentage reduction of the power-delay product of the proposed design over the reference

design is plotted in **Error! Reference source not found.**. A similar trend of percentage education compared to the area-delay product with increasing α is observed in **Error! Reference source not found.**.

Table 4.3 Comparison of Total Power and Leakage Power.

n	k	α	Total Power (μW)		% δ	Leakage (μW)		% δ
			This	Ref		This	Ref	
8	0	1	6668	6057	-10.09	276.20	277.26	0.38
	0	2	5713	5837	2.12	239.01	284.55	16.00
	0	4	4870	4371	-11.42	235.32	215.67	-9.11
	4	1	5920	6605	10.37	273.55	309.46	11.60
	4	2	6056	6492	6.72	274.09	298.81	8.27
	4	4	4960	5773	14.08	237.53	281.41	15.59
	8	1	8886	6655	-33.52	406.51	304.68	-33.42
	8	2	6787	7140	4.94	301.46	340.51	11.47
12	8	4	4462	5688	21.55	199.01	254.15	21.70
	0	1	9994	10167	1.70	418.65	493.06	15.09
	0	3	7448	7668	2.87	322.25	393.02	18.01
	0	6	6072	7737	21.52	319.09	371.00	13.99
	6	1	10614	9166	-15.80	466.99	486.23	3.96
	6	3	7294	7476	2.43	331.71	374.55	11.44
	6	6	6118	8272	26.04	341.90	419.07	18.41
	12	1	11128	9462	-17.61	508.03	467.63	-8.64
16	12	3	10466	10706	2.24	485.23	491.45	1.27
	12	6	6974	7524	7.31	351.82	358.51	1.87
	0	1	10613	11641	8.83	496.82	580.13	14.36
	0	4	10289	8982	-14.55	479.31	499.51	4.04
	0	8	8406	9022	6.83	410.80	472.98	13.15
	8	1	12900	11607	-11.14	592.01	642.28	7.83
	8	4	10138	10570	4.09	496.21	521.05	4.77
	8	8	9859	10003	1.44	513.72	547.62	6.19
20	16	1	13951	12574	-10.95	698.29	594.34	-17.49
	16	4	11112	11179	0.60	546.52	593.05	7.85
	16	8	8327	12219	31.85	423.75	621.20	31.79
	0	1	12443	13505	7.86	621.84	723.08	14.00
	0	5	10014	10819	7.44	540.74	608.48	11.13
	0	10	5524	11248	50.89	280.94	617.51	54.50
	10	1	13731	13550	-1.34	696.94	739.85	5.80
	10	5	13059	11417	-14.38	604.41	633.93	4.66
	10	10	8392	11537	27.26	512.25	672.52	23.83
	20	1	19259	12896	-49.34	954.82	722.43	-32.17
	20	5	14786	11808	-25.22	792.71	649.07	-22.13
	20	10	9837	11297	12.92	530.65	613.53	13.51

Figure 4.12 Percentage reduction of power-delay product.

4.4.3 Comparison with Removal of One Modulus Method

The dynamic range of S_1 can also be reduced to $2n+k$, $2n$ or $2n+k$ bits by removing one of the 2^n-1 , 2^{n+k} or 2^n+1 modulus channel at the expense of having larger modular operators and reduced parallelism than using the proposed base transformation method for the same reduction. Unlike the removal of a modulus channel, which is very restrictive on the target dynamic range, the proposed base transformation method allows any desired extent of range reduction to be achieved easily by selecting the appropriate α , since $\alpha \in [1, n-2]$.

Figure 4.13(a) shows the typical architecture of a three-moduli set RNS-based inner product step processor. The dynamic ranges of Datapaths 1 and 2 are M and M' , respectively with $M > M'$. Both datapaths are composed of multiply-accumulate (MAC) units. Each MAC unit includes a modular multiplier and a modular adder. Figures 4.13(b) and (c) depict the inner product step processor architectures for the proposed base transformation method and removal of one modulus channel method, respectively, for achieving the same dynamic range reduction. The hardware costs of the three architectures are evaluated using unit-gate analysis. Since the hardware costs of the binary-to-residue converter and Datapath 1 are identical for the three architectures, they are omitted in the comparison. Datapath 2 includes mod 2^n-1 adder, mod 2^{n+k} adder, mod 2^n+1 adder, mod 2^n-1 multiplier, mod 2^{n+k} multiplier and mod 2^n+1 multiplier. The mod 2^n-1 adder, mod 2^n+1 adder, mod 2^n-1 multiplier and mod 2^n+1 multiplier are

assumed to be implemented with the designs in [Che09], [Ver02], [Pie10a] and [Efs11], respectively, and their unit-gate delays and areas can be excerpted from these papers. The mod 2^{n+k} adder and multiplier do not require end-around carry propagation and correction constants. They are typically implemented with similar efficiency as their binary counterparts by discarding all bits of weight greater than 2^{n+k} in the parallel prefix and carry-save additions. In this analysis, the mod 2^{n+k} adder is implemented using Ladner-Fischer parallel prefix adder tree while the mod 2^{n+k} multiplier is implemented using Dadda tree adder. The unit-gate delay and area of mod 2^{n+k} adder are estimated to be $2\lceil\log_2(n+k)\rceil+3$ and $1.5(n+k)\lceil\log_2(n+k)\rceil+6(n+k)$ units, respectively while the unit-gate delay and area of mod 2^{n+k} multiplier are estimated to be $2\lceil\log_2(n+k)\rceil+D(n+k)+4$ and $1.5(n+k)\lceil\log_2(n+k)\rceil+4n^2+13n+15$ units, respectively, where $D(n+k)$ is the Dadda's tree depth function (see Table II of [Efs11]). Let RM-1, RM-2 and RM-3 denote the methods of removing 2^n-1 , 2^{n+k} and 2^n+1 modulus channels, respectively. The residue-to-binary conversion of RM-2, with x_1 and x_3 as the remaining residue digits, can be implemented using the proposed method for the computation of X_{13} . Its unit-gate delay and area are $2\lceil\log_2 n-1\rceil+7$ and $1.5n\lceil\log_2 n\rceil+17n-1$ units, respectively. As there is no existing design for the residue-to-binary converters of RM-1 and RM-3, they are assumed to be implemented by the two-moduli MRC technique in Equation (2.12) (see Section 2.1.3 of Chapter 2). The unit-gate delay D_{RM-1} and the unit-gate area A_{RM-1} of the residue-to-binary converter for RM-1 are estimated as follows:

$$D_{RM-1} = \begin{cases} 2\lceil\log_2 n\rceil+9 & \text{if } k=0 \\ 2\lceil\log_2 n\rceil+13 & \text{otherwise} \end{cases} \quad (4.36)$$

$$A_{RM-1} = \begin{cases} 1.5n\lceil\log_2 n\rceil+17n-2 & \text{if } k=0 \\ 1.5n\lceil\log_2 n\rceil+24n-2 & \text{otherwise} \end{cases} \quad (4.37)$$

Similarly, the unit-gate delay D_{RM-3} and the unit-gate area A_{RM-3} of the residue-to-binary converter for RM-3 are estimated as follows:

$$D_{RM-3} = \begin{cases} 2\lceil\log_2 n-1\rceil+5 & \text{if } k=0 \\ 2\lceil\log_2 n-1\rceil+9 & \text{otherwise} \end{cases} \quad (4.38)$$

$$A_{RM-3} = \begin{cases} 1.5n \lceil \log_2 n \rceil + 10n - 1 & \text{if } k = 0 \\ 1.5n \lceil \log_2 n \rceil + 17n - 1 & \text{otherwise} \end{cases} \quad (4.39)$$

Figure 4.13 Architectures of RNS-based inner product step processor (a) dynamic range reduction unexploited (b) dynamic range reduced by the proposed base transformation (c) dynamic range reduced by the removal of one modulus channel.

Figure 4.14 depicts, in percentage, the reduction in the area-delay product (δ_{ADP}) achieved by using the proposed design and the removal of one modulus channel methods to reduce the dynamic range from $M = 36$ bits (with $n = 12$ and $k = 0$) of Datapath 1 to three different dynamic ranges $M' = 27, 24$ and 21 bits of Datapath 2. Datapath 2 of 10, 15 and 20 MAC units are considered. Since it is not possible to reduce the dynamic range from 36 bits to 27 bits by removing any modulus channel, only the δ_{ADP} for the proposed method is shown for $M' = 27$. The proposed method has the highest δ_{ADP} in all cases, with δ_{ADP} steadily increasing with the number of MAC units. On the contrary, δ_{ADP} due to the removal of any modulus channel declines slightly with the increasing number of MAC units. This shows that the proposed

method can better exploit the dynamic range reduction than the removal of modulus channel, improving further the area-time efficiency of the overall system.

Figure 4.14 The percentages of reduction on area-delay product by using the proposed design (proposed), RM-1, RM-2, and RM-3 to reduce the dynamic range from $M = 36$ bits to $M' = 27$, 24 and 21 bits with datapath of $N = 10, 15$, and 20 MAC units.

4.5 Summary and Discussions

Reckoning the need to change the residue arithmetic unit's precision in application-specific computations to tap the full potential of the RNS, this chapter presents an elegant algorithm and its efficient hardware implementation to enable base transformation in the RNS. It directly maps the residues from a signed RNS of a three-moduli set $\{2^n - 1, 2^{n+k}, 2^n + 1\}$ to another three-moduli set $\{2^{n-\alpha} - 1, 2^{n+k-\alpha}, 2^{n-\alpha} + 1\}$ of lower dynamic range. The proposed design complements RNS scaling to substantially reduce the sizes of modulo arithmetic operators. A clear advantage in terms of VLSI efficiency metrics over the best conceivable improvisation to realize base transformation is demonstrated. The savings in terms of area-delay product and energy consumption operating at full speed are particularly pronounced with increasing difference in the dynamic ranges of the moduli sets, which corresponds to the situation where the advantage of a single base RNS is heavily undermined when the numerical ranges of the arithmetic operands have been greatly reduced.

Since the dynamic range of S_1 is $M = 2^{n+k} (2^{2n} - 1)$ (i.e., $3n + k$ bits), there exist more than one three-moduli set S_1 with different n and k combinations that can be chosen to satisfy any dynamic range with up to one bit resolution. The dynamic ranges M and M' of S_1 and S_2 are determined by the application, which means that α is fixed. The optimal value of n and k can be selected by using the unit-gate analysis to estimate the cost of the proposed design for a given α with all combinations of n and k that fulfil M .

The proposed algorithm can be suitably adapted for higher cardinality supersets. With a larger number of moduli, there are more choices of moduli for a greater variety of dynamic ranges. The restriction of α , which must be an integer ranges from 1 to $n-2$, on the target dynamic range M' can be lifted by adding one or two coprime moduli into S_2 , i.e., $S_2 \equiv \{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1, m'_4\}$ or $\{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1, m'_4, m'_5\}$, where m'_4 and m'_5 can be in the form of $2^{n-\alpha-1}-1, 2^{n-\alpha+1}-1, 2^{2(n-\alpha)+1}-1$ or $2^{n-\alpha+1}+1$. The residues, x'_4 and x'_5 , can be computed by modulo reduction in m'_4 and m'_5 using Equation (4.11). These residue generators can be added into the residue remapping block shown in Figure 4.5. Alternatively, S_1 can also be modified by adding extra moduli of similar forms. For this case, the computation units of X_{13} and β_L needs to be redesigned based on the same techniques discussed in Section 4.4 to simplify its implementation.

In the next chapter, different range division and detection strategies for single and multiple residue digit errors will be investigated. The nature of the problem and the approach to it are fundamentally different as arbitrary moduli with limited number theoretic properties are involved in the inter-modular operations.

Chapter 5

Non-Iterative Single and Multiple Residue Error Detection and Correction Algorithms

5.1 Introduction

Error detection and correction capability of RRNS has been utilized in several fault-tolerant applications reported in literature [Yan02], [How06], [Etz80], [Bec93], [Kel00], [Har11], [Han02], [Lie06], [Zha12], [Avi13], [Yat13]. In [Etz80], a sixth-order band reject filter was implemented in RRNS. Besides error correction, overflow can be detected by adding two redundant moduli. RRNS was also used to develop a fault-tolerant convolution algorithm that is well suited for the implementation on multiprocessor systems [Bec93]. By using long division method, the algorithm can detect and correct the processing errors. In order to curb the frequency-selective fading problem, the symbol-by-symbol adaptive multicarrier modulation is coded in RRNS [Kel00]. According to the benchmark, the proposed method has a better channel signal-to-noise ratio of 15 dB over the convolutional constituent code. In addition, an error correction code based on six-moduli RRNS was implemented on hybrid memories [Har11]. Compared to Reed-Solomon codes, this code provides larger data storage with similar error correction capability.

Typically, error detection and correction in RRNS are performed in three sequential steps: error existence check, identification of erroneous residue digits and error correction, among which identification of erroneous residue digits is the most difficult task. Error is normally detected by checking if the magnitude of the residue representation falls outside the legitimate computation range. The existing single residue error detection and correction algorithms that use modulus projection [Tan10], [Tin04], [Wan95], base extension [Kri92], [Cla93], range detection [Su90], etc. differ by the ways error location is identified from the received residues. These methods are either complicated, iterative, involve large modulo operations or require a lot of stored tables.

Upon extracting the error position and magnitude, the error can be corrected by either subtracting the error digit in the residue domain or computing the magnitude of error-free residue representation. On the other hand, detecting and correcting multiple residue digit errors are more challenging and time consuming than single residue digit error. This is mainly due to the enormous number of combinations of different residue digit locations and magnitudes to look for errors in the second step. In general, three different methods are utilized by the existing multiple residue error detection and correction algorithms in locating the erroneous residue digits. They are consistency check using syndrome [Yau73], [Sun92], error recovery by the CRT [Man76], [Gol00] and modulus projection [Sun92], [Goh08]. These algorithms require iterative or recursive computations and comparisons, which are not amenable to parallel hardware implementation.

In this chapter, two new algorithms are proposed for correcting single and multiple residue digit errors. The proposed single residue error detection and correction algorithm distinguishes itself from others by the proposed syndrome to identify the erroneous residue and decode its error value. One important contribution is the proof of syndrome uniqueness, which makes it possible to create an error lookup table indexed by the syndrome for every possible residue digit error. The uniqueness criteria are shown to be easily fulfilled by many choices of redundant moduli from an abundant number of coprime integers and the circuit implementation of the proposed algorithm is also suggested. On the other hand, the proposed multiple residue error detection and correction algorithm divides the residue digits of RRNS into three groups so that any combination of residue digit errors can be categorically identified by one of the seven error location categories. Three syndromes are derived from the received residue representation to detect up to $2t$ and correct up to t residue digit errors, where the number of redundant moduli is $2t$. Irrespective of the number of moduli, the residue representation of errors can be uniquely retrieved by these syndromes from six lookup tables in a few steps and subtracted directly from the received residue digits to correct the errors. The latency is fixed and independent of the size of the moduli set. The second moduli selection criterion that imposes a high redundant space can be completely eliminated by making the number of syndrome computations dependent on the number of information moduli. Comparing with the existing algorithms, the proposed

algorithm is simpler and have very low error decoding complexity. Only one moderate size modulo subtraction is required for each syndrome computation, which can be decomposed into smaller modulo subtractors for efficient implementation. The proposed algorithm categorizes the different combinations of residue digit errors into seven location groups with no restriction on the number of information and redundant moduli, which is in stark contrast to the proposed single residue error detection and correction algorithm which locates a single residue digit error for RRNS with only two redundant moduli.

This rest of this chapter is divided into three main sections. The proposed single and multiple residue error detection and correction algorithms are presented in two large sections, Sections 5.2 and 5.3, respectively while Section 5.4 summarizes the chapter. The sub-sections of Section 5.2 are organized as follows. In Section 5.2.1, the proof of uniqueness of the proposed algorithm is shown. The proposed procedure and architecture are presented in Section 5.2.2 and compared to the existing algorithms in Section 5.2.3. Section 5.3 is constructed as follows. The proposed algorithm, including the syndrome derivation, error detection and correction procedure, an illustrative example and the uniqueness proof are presented in Sections 5.3.1, 5.3.2, 5.3.3 and 5.3.4, respectively. An extended version with variable number of syndromes for redundant space reduction is proposed in Section 5.3.5. In Section 5.3.6, the proposed and extended algorithms are compared against the existing algorithms.

5.2 Single Residue Error Detection and Correction

5.2.1 Uniqueness of Proposed Error Syndrome

Let e_i be the error digit introduced into the modulus channel m_i for $1 \leq i \leq k+2$. The error magnitude E_i can be represented by an n -tuple $(0, \dots, e_i, \dots, 0)$ in RRNS, where $n = k + 2$. Using the CRT, E_i can be computed as [Kri92]:

$$E_i = (M_N / m_i) \left\| M_i^{-1} \right\|_{m_i} e_i \Big|_{m_i} = a_i M_N / m_i = a_i M_K M_R / m_i \quad (5.1)$$

where $a_i = \left\| M_i^{-1} \right\|_{m_i} e_i \Big|_{m_i}$ is an integer in the range $[1, m_i - 1]$.

The relationship between a residue representation $\tilde{X} \equiv (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n)$ that is contaminated with a residue digit error $E_i \equiv (0, \dots, e_i, \dots, 0)$ and an error-free residue representation, $X \equiv (x_1, x_2, \dots, x_n)$ is given by

$$\begin{aligned} (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n) &\equiv (x_1, x_2, \dots, x_n) + (0, \dots, e_i, \dots, 0) \\ &= (x_1, \dots, |x_i + e_i|_{m_i}, \dots, x_n) \end{aligned} \quad (5.2)$$

E_i is unique for every e_i value in m_i . If E_i can be extracted from the residue representation of $\tilde{X}$, the erroneous residue digit $\tilde{x}_i$ can be identified and corrected by $x_i = |\tilde{x}_i - e_i|_{m_i}$. As RRNS is a non-weighted number system, the error magnitude cannot be easily recovered from its residue representation. Decoding the received residue representation by the CRT requires large modulo operations and is inefficient. To extract the error magnitude from an erroneous residue representation without involving large modulo operation, a syndrome δ is defined as follows:

$$\delta = \left| \tilde{X}|_{M_K} - \tilde{X}|_{M_R} \right|_{M_R} \quad (5.3)$$

where $|\tilde{X}|_{M_K} \equiv (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$ and $|\tilde{X}|_{M_R} \equiv (\tilde{x}_{k+1}, \tilde{x}_{k+2})$.

Theorem 5.1: When there is no residue digit error, δ is zero. If there is any error in the information (or redundant) residue, δ is modulo M_R congruent to $E_i \bmod M_K$ (or M_R) with or without an offset of M_K (or M_R).

Proof: If $\tilde{X} \equiv (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n)$ is error-free, then $|\tilde{X}|_{M_K} = |X|_{M_K} = X$ and $|\tilde{X}|_{M_R} = |X|_{M_R}$.

$$\delta = \left| X - |X|_{M_R} \right|_{M_R} = \left| \lfloor X/M_R \rfloor M_R \right|_{M_R} = 0 \quad (5.4)$$

To use the syndrome δ to correct a residue digit error, δ must be unique for every possible error digit. The error digit can appear in either an information residue or a redundant residue. If an information residue is erroneous, then $|\tilde{X}|_{M_K} = |X + E_i|_{M_K}$ and $|\tilde{X}|_{M_R} = |X|_{M_R}$.

$$\begin{aligned} \delta &= \left| |X + E_i|_{M_K} - |X|_{M_R} \right|_{M_R} \\ &= \left| |X + |E_i|_{M_K} - \alpha_K M_K - |X|_{M_R} \right|_{M_R} = \left| |E_i|_{M_K} - \alpha_K M_K \right|_{M_R} \end{aligned} \quad (5.5)$$

where $\alpha_K = 0$ if $|X + E_i|_{M_K} < M_K$ and $\alpha_K = 1$ otherwise.

If a redundant residue is erroneous, then $|\tilde{X}|_{M_K} = X$ and $|\tilde{X}|_{M_R} = |X + E_i|_{M_R}$.

$$\begin{aligned} \delta &= |X - |X + E_i|_{M_R}|_{M_R} \\ &= |X - (|X|_{M_R} + |E_i|_{M_R} - \alpha_R M_R)|_{M_R} = |-|E_i|_{M_R}|_{M_R} \end{aligned} \quad (5.6)$$

where $\alpha_R = 0$ if $|X|_{M_R} + |E_i|_{M_R} < M_R$ and $\alpha_R = 1$ otherwise.

Theorem 5.2: The syndrome δ is distinct for every possible error digit provided that (1) $M_R > 2\sum_{j=1}^k (m_j - 1) + m_{k+1} + m_{k+2} - 2$ and (2) the product of the redundant moduli must be at least two times larger than the product of the two largest information moduli, i.e., $M_R > 2 \times \max(m_i m_j)$.

Proof: The total number of possible single residue error in the information residue channels and redundant residue channels can be computed as $\sum_{j=1}^k (m_j - 1)$ and $(m_{k+1} - 1) + (m_{k+2} - 1)$, respectively. There are $\sum_{j=1}^k (m_j - 1)$ different values of e_i in the information channels. According to Equation (5.5), the magnitude of any residue error can be decoded from either $\delta = |E_i|_{M_K}|_{M_R}$ or $\delta = |E_i|_{M_K} + M_K|_{M_R}$ depending on the value of α_K . According to (5.6), any of the $(m_{k+1} - 1) + (m_{k+2} - 1)$ possible error magnitudes in the redundant channels can be decoded directly from $\delta = |-|E_i|_{M_R}|_{M_R}$. Since $\delta \in [0, M_R - 1]$, condition (1) in *Theorem 5.2* can be fulfilled by having

$$M_R > 2\sum_{j=1}^k (m_j - 1) + m_{k+1} + m_{k+2} - 2 \quad (5.7)$$

To prove the second criterion, let E_i and E_j ($E_i > E_j$) be two different error magnitudes due to the residue errors in modulus channels m_i and m_j , respectively, and m_i needs not be larger than nor different from m_j . If $m_i = m_j$, E_i and E_j are two error magnitudes due the residue errors in the same modulus channel. In any case, δ_i due to E_i must be distinguishable from δ_j due to E_j .

Case 1: E_i and E_j are due to different residue errors in any information residues (i.e., $i, j \in [0, k]$).

From Equation (5.5), $\delta_i = \left| E_i \right|_{M_K} - \alpha_{Ki} M_K \Big|_{M_R}$ and $\delta_j = \left| E_j \right|_{M_K} - \alpha_{Kj} M_K \Big|_{M_R}$. For this case,

$\delta_i \neq \delta_j$ if and only if $\Delta = \left| E_i \right|_{M_K} - \alpha_{Ki} M_K - \left(\left| E_j \right|_{M_K} - \alpha_{Kj} M_K \right)$ is not a multiple of M_R .

Substituting $E_i = a_i M_K M_R / m_i$ from Equation (5.1) into $\Delta = \left| E_i \right|_{M_K} - \alpha_{Ki} M_K - \left| E_j \right|_{M_K} + \alpha_{Kj} M_K =$

$E_i - \beta_{Ki} M_K - \alpha_{Ki} M_K - E_j + \beta_{Kj} M_K + \alpha_{Kj} M_K$, Δ can be expressed as:

$$\Delta = \left[\frac{a_i M_K}{m_i} - \frac{a_j M_K}{m_j} \right] M_R - (\alpha_{Ki} + \beta_{Ki} - \alpha_{Kj} - \beta_{Kj}) M_K \quad (5.8)$$

where $\beta_{Ki} = \lfloor E_i / M_K \rfloor$ and $\beta_{Kj} = \lfloor E_j / M_K \rfloor$.

Since $a_i M_K / m_i$ and $a_j M_K / m_j$ are integers for the information channel errors. The expression

of Equation (5.8) is not a multiple of M_R if $\alpha_{Ki} + \beta_{Ki} - \alpha_{Kj} - \beta_{Kj}$ is non-zero, i.e.,

$$\left| \alpha_{Ki} + \beta_{Ki} - \alpha_{Kj} - \beta_{Kj} \right| \geq 1 \quad (5.9)$$

Assuming that $\alpha_{Ki} + \beta_{Ki} > \alpha_{Kj} + \beta_{Kj}$, then

$$\begin{aligned} \alpha_{Ki} + \beta_{Ki} - \alpha_{Kj} - \beta_{Kj} &\geq 1 \\ \lfloor E_i / M_K \rfloor - \lfloor E_j / M_K \rfloor + (\alpha_{Ki} - \alpha_{Kj}) &\geq 1 \\ E_i / M_K - E_j / M_K + (\alpha_{Ki} - \alpha_{Kj}) &> 1 \\ (a_i / m_i - a_j / m_j) M_R &> 1 + \alpha_{Kj} - \alpha_{Ki} \\ M_R &> \frac{(1 + \alpha_{Kj} - \alpha_{Ki}) m_i m_j}{a_i m_j - a_j m_i} \end{aligned} \quad (5.10)$$

Since $a_i < m_i$, $a_j < m_j$ and both m_i and m_j are coprime, $\min(a_i m_j - a_j m_i) = 1$. Also, $\max(1 + \alpha_{Kj}$

$- \alpha_{Ki}) = 2$ because $\alpha_{Ki}, \alpha_{Kj} \in \{0, 1\}$. Thus, the minimum value that M_R must exceed is

$$M_R > 2 \times \max(m_i m_j) \quad (5.11)$$

In other words, the product of the two redundant moduli must be greater than twice the product of the two largest information moduli. The same conclusion can also be drawn for $\alpha_{K_i} + \beta_{K_i} < \alpha_{K_j} + \beta_{K_j}$.

Case 2: E_i and E_j are due to different errors in any redundant residues (i.e., $i, j \in \{k+1, k+2\}$).

From Equation (5.6),

$$\begin{aligned}\delta_i &= \left| -|E_i|_{M_R} \right|_{M_R} = M_R - |E_i|_{M_R} = M_R - (E_i - \beta_{Ri} M_R) \\ &= M_R - (a_i M_K M_R / m_i - \beta_{Ri} M_R) \\ &= [(\beta_{Ri} + 1)m_i - a_i M_K] (M_R / m_i) = Z_i (M_R / m_i)\end{aligned}\tag{5.12}$$

where $\beta_{Ri} = \lfloor E_i / M_R \rfloor$ and $Z_i = (\beta_{Ri} + 1)m_i - a_i M_K$.

From Equation (5.12), δ_i is an integer multiple of M_R / m_i . Since $M_R = m_{k+1} \times m_{k+2}$, δ_{k+1} and δ_{k+2} are the integer multiples of m_{k+2} and m_{k+1} , respectively. As m_{k+1} and m_{k+2} are coprime, δ_{k+1} and δ_{k+2} are always distinguishable.

For δ_i and δ_j due to different error digits in the same redundant channel (i.e., $m_i = m_j$) to be distinguishable, $Z_i \neq Z_j$. Since m_i and M_K are coprime and $a_i - a_j \in [-m_i + 2, m_i - 2]$,

$$\begin{aligned}(\beta_{Ri} - \beta_{Rj})m_i &\neq (a_i - a_j)M_K \\ (\beta_{Ri} + 1)m_i - a_i M_K &\neq (\beta_{Rj} + 1)m_i - a_j M_K \Rightarrow Z_i \neq Z_j\end{aligned}\tag{5.13}$$

From Equation (5.13), the syndromes due to different error digits in the same redundant channel are unconditionally distinguishable.

Case 3: E_i and E_j are due to errors in information and redundant residues, respectively (i.e., $i \in [0, k]$ and $j \in \{k+1, k+2\}$).

From Equations (5.5) and (5.6), $\delta_i = \left| |E_i|_{M_K} - \alpha_{Ki} M_K \right|_{M_R}$ and $\delta_j = \left| -|E_j|_{M_R} \right|_{M_R}$. For this case,

$\delta_i \neq \delta_j$ if and only if $\Delta = |E_i|_{M_K} - \alpha_{Ki} M_K - (-|E_j|_{M_R})$ is not a multiple of M_R .

$$\begin{aligned}\Delta &= E_i - \beta_{Ki} M_K - \alpha_{Ki} M_K + E_j - \beta_{Rj} M_R \\ &= (a_i M_K / m_i - \beta_{Rj}) M_R - (\beta_{Ki} + \alpha_{Ki} - a_j M_R / m_j) M_K\end{aligned}\quad (5.14)$$

where $\beta_{Ki} = \lfloor E_i / M_K \rfloor$ and $\beta_{Rj} = \lfloor E_j / M_R \rfloor$.

Since $a_i M_K / m_i - \beta_{Rj}$ is an integer, Equation (5.14) is not a multiple of M_R if and only if

$\beta_{Ki} + \alpha_{Ki} - a_j M_R / m_j$ is non-zero, i.e.,

$$|\beta_{Ki} + \alpha_{Ki} - a_j M_R / m_j| \geq 1 \quad (5.15)$$

Assume $\beta_{Ki} + \alpha_{Ki} > a_j M_R / m_j$, then

$$\begin{aligned}\beta_{Ki} + \alpha_{Ki} - a_j M_R / m_j &\geq 1 \\ \lfloor a_i M_R / m_i \rfloor + \alpha_{Ki} - a_j M_R / m_j &\geq 1 \\ a_i M_R / m_i + \alpha_{Ki} - a_j M_R / m_j &> 0 \\ M_R &> \frac{-\alpha_{Ki} m_i m_j}{a_i m_j - a_j m_i}\end{aligned}\quad (5.16)$$

When $a_i m_j - a_j m_i = -1$ and $\alpha_{Ki} = 1$, $M_R > \max(m_i \times m_j)$, i.e.,

$$\begin{aligned}m_{k+1} \cdot m_{k+2} &> \max(m_i) \cdot \max(m_{k+1}, m_{k+2}) \\ \min(m_{k+1}, m_{k+2}) &> \max(m_i)\end{aligned}\quad (5.17)$$

The inequality (5.17) suggests that the smallest redundant modulus must be greater than the largest information modulus. The same conclusion can be drawn if $\beta_{Ki} + \alpha_{Ki} < a_j M_R / m_j$.

Combining the inequalities, (5.11) for Case 1 and (5.17) for Case 3, the most stringent condition is $M_R > 2 \times \max(m_i m_j)$.

5.2.2 Proposed Algorithm and Architecture for Single Residue Digit Error

Correction

Based on *Theorems 5.1* and *5.2*, a new single residue digit error correction method is proposed as follows.

- (1) Compute $|\tilde{X}|_{M_K}$ and $|\tilde{X}|_{M_R}$ by evaluating the magnitudes of $(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$ and $(\tilde{x}_{k+1}, \tilde{x}_{k+2})$, respectively.
- (2) Compute $\delta = \left| |\tilde{X}|_{M_K} - |\tilde{X}|_{M_R} \right|_{M_R}$.
- (3) If $\delta = 0$, there is no error. Stop.
- (4) Otherwise, use δ as the address to an error detection table to identify the erroneous modulus channel m_i and error digit e_i .
- (5) Correct the erroneous residue digit $\tilde{x}_i$ by $x_i = |\tilde{x}_i - e_i|_{m_i}$.

Example 5.1: Consider the moduli set $\{7, 9, 11, 13, 17\}$, where 13 and 17 are the redundant moduli. Since $M_R = 13 \times 17 = 221 > 2(6+8+10)+13+17-2 = 76$ and $M_R = 221 > 2 \times 9 \times 11 = 198$, conditions (1) and (2) of *Theorem 5.2* are fulfilled. Let $\tilde{X} \equiv 68297 \equiv (5, 5, 9, 8, 8)$. $|\tilde{X}|_{M_K} = 383$ and $|\tilde{X}|_{M_R} = 8$ can be recovered from the residues $(5, 5, 9)$ of RNS $\{7, 9, 11\}$ and $(8, 8)$ of RNS $\{13, 17\}$, respectively. The syndrome $\delta = \left| |\tilde{X}|_{M_K} - |\tilde{X}|_{M_R} \right|_{M_R} = |383 - 8|_{221} = 154$. Since δ is non-zero, one of the residue digits is erroneous. The lookup table indexed by 154 returns $e_i = 1$ and $i = 2$ signifying $e_2 = 1$. Thus $x_2 = |\tilde{x}_2 - e_2|_{m_2} = |5 - 1|_9 = 4$. The corrected residue representation is $(5, 4, 9, 8, 8) \equiv X = 229$.

The architecture for the syndrome computation, error detection and correction is shown in Figure 5.1. The k -residue and two-residue CRTs evaluate $|\tilde{X}|_{M_K}$ and $|\tilde{X}|_{M_R}$ from $(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$ and $(\tilde{x}_{k+1}, \tilde{x}_{k+2})$, respectively. $|\tilde{X}|_{M_R}$ is then subtracted from $|\tilde{X}|_{M_K}$ by a mod M_R subtractor to produce the syndrome δ as the address to an error lookup table. The lookup table stores the corresponding $\lceil \log_2 m_{k+2} \rceil$ -bit residue error digit e_i and $\lceil \log_2(k+2) \rceil$ -bit erroneous modulus identifier i , which can be computed by using Equations (5.5) and (5.6). The comparator of the error correction module that compares i with a fixed integer can be implemented using simple logic gates. e_i is then gated into the i -th modular subtractor of the RRNS by the output of the i -

th comparator. The outputs of the $k+2$ subtractors are the error-free residue digits $(x_1, x_2, \dots, x_k, x_{k+1}, x_{k+2})$.

Figure 5.1 Architecture for the proposed residue error correction algorithm.

5.2.3 Comparison

Table 5.1 shows the comparison of existing single residue error correction algorithms in terms of the number of redundant moduli r , the need for iterative consistency check, the output representation and the most stringent criterion on their moduli. All but [Pon08] and [Kat96] require redundant moduli. Redundancy is indirectly introduced into the legitimate dynamic range of [Pon08] by scaled RNS and [Kat96] by moduli with common factors. For comparable legitimate range, these two algorithms have larger moduli and hence slower residue arithmetic operations. No iterative consistency check is required by [Tan10], [Cla93], [Su90], [Pon08] and the proposed algorithm to detect residue error, making them more amenable to hardware implementation. The corrected output of the proposed algorithm is represented in residue domain, which is useful for correcting intermediate arithmetic processing and data transfer error within the RNS. Unlike [Cla93], which must have an even number of moduli, there is no restriction on the number of moduli for the proposed algorithm. Simialr to [Su90], the syndrome

uniqueness criteria imposed on the moduli of the RRNS for the proposed algorithm is easy to fulfil due to the abundant choice of coprime moduli.

Table 5.1 Comparison of Single Residue Error Correction Algorithms.

Algorithm	r	Iterative?	Output domain	Most stringent moduli selection criterion
[Tan10]	2	No	integer	$M_K \approx M_R$
[Tin04]	2	Yes	integer	$m_{k+1}, m_{k+2} > m_i \forall i \leq k$
[Wan95]	≥ 3	Yes	integer	$M_R \geq \max(m_p, m_q, m_s), 1 \leq p, q, s \leq k+r$
[Kri92]	2	Yes	residue	$M_R > 2m_i m_j - m_i - m_j, 1 \leq i, j \leq k$
[Cla93]	2	No	residue	even n ; $m_{k+2} < \min(m_i, m_j)$
[Su90]	2	No	residue	$M_R > 2m_i m_j - m_i - m_j, 1 \leq i, j \leq k$
[Pon08]	0	No	integer	$m > \max(m_i, m_j)$
[Kat96]	0	Yes	residue	$\gcd(m_i, m_j) \neq 1$
This	2	No	residue	$M_R > 2 \times \max(m_i, m_j)$

5.3 Multiple Residue Error Detection and Correction

Let l_i be the positions of the erroneous residue digits and e_{l_i} be the magnitude of error introduced into the corresponding residue digit in the modulus channel m_{l_i} for $1 \leq i \leq t$. Assume that t residue digit errors are introduced into $X \equiv (x_1, x_2, \dots, x_n)$, the error magnitude E_l can be represented by an n -tuple $(e_{l_1}, \dots, 0, e_{l_2}, \dots, e_{l_t})$ in RRNS, where $n = k + r$ and $r = 2t$. The residue representation of X' with t residue digit errors can be expressed as:

$$\begin{aligned}
 X' &= X + E_l \\
 (x'_1, x'_2, \dots, x'_n) &\equiv (x_1, x_2, \dots, x_n) + (e_{l_1}, \dots, 0, e_{l_2}, \dots, e_{l_t}) \\
 &\equiv \left(|x_{l_1} + e_{l_1}|_{m_{l_1}}, \dots, |x_{l_t} + e_{l_t}|_{m_{l_t}} \right)
 \end{aligned} \tag{5.18}$$

For convenience, the residue representation of E_l is referred to as the error vector henceforth. Similar to the computation of X in Equation (2.14) (see Section 2.1.3 of Chapter 2), E_l can also be computed from its error vector $(e_{l_1}, \dots, 0, e_{l_2}, \dots, e_{l_t})$ by the CRT:

$$\begin{aligned}
 E_l &= \left| \sum_{i=1}^t \frac{M_N}{m_{l_i}} \left\| M_{l_i}^{-1} \right\|_{m_{l_i}} e_{l_i} \right\|_{m_{l_i}} \Big|_{M_N} = \sum_{i=1}^t a_{l_i} \frac{M_N}{m_{l_i}} - r_n M_N \\
 &= a \frac{M_N}{\prod_{i=1}^t m_{l_i}} = a \prod_{\substack{j=1 \\ l_j \neq l_i}}^{n-t} m_{l_j}
 \end{aligned} \tag{5.19}$$

where $|M_{l_i}|_{m_i} = |M_N/m_{l_i}|_{m_i}$ and $|M_{l_i}^{-1}|_{m_i}$ is its multiplicative inverse. $a_{l_i} = \left| |M_{l_i}^{-1}|_{m_i} e_{l_i} \right|_{m_i}$, r_n is an

integer smaller than n , and a is an integer in the range $\left[1, \prod_{i=1}^t m_{l_i} - 1 \right]$.

From Equation (5.19), E_l is a multiple of the product of error-free moduli. When $m_{k+2}, \dots, m_{k+r} > m_i \forall i \leq k$ and $t \leq \lfloor r/2 \rfloor$, the minimum difference between the error magnitudes of any two error vectors is always greater than M_K . Since $X < M_K$, an erroneous residue representation can be detected by checking if X' falls in the illegitimate range. However, decoding X' from $(x'_1, x'_2, \dots, x'_n)$ by the CRT requires large compute-intensive modulo M_N operations. Upon detecting the errors, the locations of the erroneous residues have to be identified before they can be corrected. Conventionally, the identification of the locations of erroneous residues involves time-consuming iterative computations which escalate with the size of the chosen moduli set. In order to overcome the aforementioned shortcomings, a new syndrome computation method is proposed in this section. It uses smaller modulo operations and the number of computations is fixed regardless of the size of moduli set.

5.3.1 Derivation of Syndromes for Error Identification

From Equation (5.19), E_l is bounded by M_N . Thus, each error vector corresponds to a unique E_l . Given the error magnitude E_l , the corresponding error digits e_{l_i} and error locations l_i can be determined unambiguously. Unfortunately, E_l cannot be obtained by directly decoding the residue representation $(x'_1, x'_2, \dots, x'_n)$ because X is unknown.

To avoid the use of large modulo M_N operations for error decoding, the n residue digits $(x'_1, x'_2, \dots, x'_n)$ of X' are divided into three distinct groups, $(x'_1, x'_2, \dots, x'_k)$, $(x'_{k+1}, x'_{k+2}, \dots, x'_{k+t})$ and $(x'_{k+t+1}, x'_{k+t+2}, \dots, x'_{k+r})$, such that the modulo operations required to unambiguously resolve the error vector are of sizes approximately three times smaller than M_N .

The dynamic range M_N of the RRNS $\{m_1, m_2, \dots, m_k, m_{k+1}, \dots, m_{k+r}\}$, where $m_{k+1}, m_{k+2}, \dots, m_{k+r} > m_i \forall i \leq k$, can then be divided into

$$\begin{aligned} M_N &= \prod_{i=1}^k m_i \times \prod_{i=k+1}^{k+r} m_i = \prod_{i=1}^k m_i \times \prod_{i=k+1}^{k+t} m_i \times \prod_{i=k+t+1}^{k+r} m_i \\ &= M_K \times M_U \times M_V \end{aligned} \quad (5.20)$$

where $M_K = \prod_{i=1}^k m_i$ is the product of the information moduli, $M_U = \prod_{i=k+1}^{k+t} m_i$ is the product of the first t redundant moduli and $M_V = \prod_{i=k+t+1}^{k+r} m_i$ is the product of the remaining t redundant moduli.

For ease of exposition, the following notations are defined.

- a) X denotes the magnitude of the error-free RRNS representation $(x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_{k+r})$. X falls in the range $[0, M_K - 1]$.
- b) $\tilde{X}$ denotes the magnitude of the received RRNS representation $(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k, \tilde{x}_{k+1}, \dots, \tilde{x}_{k+r})$. $\tilde{X}$ falls in the range $[0, M_N - 1]$.
- c) $\tilde{X}_K$, $\tilde{X}_U$ and $\tilde{X}_V$ denote the magnitudes of the RNS representations of the received information residue digits, the first t received redundant residue digits and the remaining t received redundant residue digits, respectively, i.e., $\tilde{X}_K \equiv (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$, $\tilde{X}_U \equiv (\tilde{x}_{k+1}, \tilde{x}_{k+2}, \dots, \tilde{x}_{k+t})$ and $\tilde{X}_V \equiv (\tilde{x}_{k+t+1}, \tilde{x}_{k+t+2}, \dots, \tilde{x}_{k+r})$. $\tilde{X}$, $\tilde{X}_K$, $\tilde{X}_U$ and $\tilde{X}_V$ may be erroneous or error free.
- d) E_K , E_U and E_V denote the magnitudes of the error digits located in the information modulus channels, the first t redundant modulus channels and the remaining t redundant modulus channels, respectively, i.e., $E_K \equiv (e_{k_1}, e_{k_2}, \dots, e_{k_l}, 0, \dots, 0)$, $E_U \equiv (0, \dots, 0, e_{u_1}, e_{u_2}, \dots, e_{u_j}, 0, \dots, 0)$ and $E_V \equiv (0, \dots, 0, e_{v_1}, e_{v_2}, \dots, e_{v_l})$, where $(k_1, k_2, \dots, k_l)$, $(u_1, u_2, \dots, u_j)$ and $(v_1, v_2, \dots, v_l)$ represent the positions of the error digits and $i, j, l \leq t$. E_K , E_U and E_V are bounded by M_N and can be computed from their corresponding error digits using Equation (5.19).

Based on the above notations, $\tilde{X}_K$, $\tilde{X}_U$, and $\tilde{X}_V$ can be expressed in term of $\tilde{X}$ by

$$\tilde{X}_K = \left| \tilde{X} \right|_{M_K} \quad (5.21)$$

$$\tilde{X}_U = \left| \tilde{X} \right|_{M_U} \quad (5.22)$$

$$\tilde{X}_V = \left| \tilde{X} \right|_{M_V} \quad (5.23)$$

If there is no error, $\tilde{X} = X$. Based on Equations (5.21)-(5.23),

$$\tilde{X}_K = \left| X \right|_{M_K} = X \quad (5.24)$$

$$\tilde{X}_U = \left| X \right|_{M_U} = X \quad (5.25)$$

$$\tilde{X}_V = \left| X \right|_{M_V} \quad (5.26)$$

It should be noted that Equation (5.25) is valid iff $M_U > M_K$ and M_U and M_K are coprime.

Otherwise, if there is(are) erroneous residue digit(s) in one or more groups among

$(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_k)$, $(\tilde{x}_{k+1}, \tilde{x}_{k+2}, \dots, \tilde{x}_{k+t})$ and $(\tilde{x}_{k+t+1}, \tilde{x}_{k+t+2}, \dots, \tilde{x}_{k+r})$, $\tilde{X}$ can be expressed as

$\tilde{X} = X + E$, i.e.,

$$\tilde{X}_K = \left| X + E_K \right|_{M_K} = X + \left| E_K \right|_{M_K} - \alpha_K M_K \quad (5.27)$$

$$\tilde{X}_U = \left| X + E_U \right|_{M_U} = X + \left| E_U \right|_{M_U} - \alpha_U M_U \quad (5.28)$$

$$\tilde{X}_V = \left| X + E_V \right|_{M_V} = \left| X \right|_{M_V} + \left| E_V \right|_{M_V} - \alpha_V M_V \quad (5.29)$$

where

$$\alpha_K = \begin{cases} 0 & \text{if } X + \left| E_K \right|_{M_K} < M_K \\ 1 & \text{if } X + \left| E_K \right|_{M_K} \geq M_K \end{cases} \quad (5.30)$$

$$\alpha_U = \begin{cases} 0 & \text{if } X + \left| E_U \right|_{M_U} < M_U \\ 1 & \text{if } X + \left| E_U \right|_{M_U} \geq M_U \end{cases} \quad (5.31)$$

$$\alpha_V = \begin{cases} 0 & \text{if } \left| X \right|_{M_V} + \left| E_V \right|_{M_V} < M_V \\ 1 & \text{if } \left| X \right|_{M_V} + \left| E_V \right|_{M_V} \geq M_V \end{cases} \quad (5.32)$$

Since the received residue digits are grouped into three distinct groups, the erroneous residue digits may exist in one of the following categories based on their error locations.

EL1: Erroneous residue digit(s) in $\tilde{X}_K$ only.

EL2: Erroneous residue digit(s) in $\tilde{X}_U$ only.

EL3: Erroneous residue digit(s) in $\tilde{X}_V$ only.

EL4: Erroneous residue digits in $\tilde{X}_K$ and $\tilde{X}_U$.

EL5: Erroneous residue digits in $\tilde{X}_K$ and $\tilde{X}_V$.

EL6: Erroneous residue digits in $\tilde{X}_U$ and $\tilde{X}_V$.

EL7: Erroneous residue digits in $\tilde{X}_K$, $\tilde{X}_U$ and $\tilde{X}_V$.

In principle, the presence of any t residue errors in the N residue digits can be detected by two syndromes, δ_1 and δ_2 . δ_1 is computed from $(x'_1, x'_2, \dots, x'_k)$ and $(x'_{k+1}, x'_{k+2}, \dots, x'_{k+t})$ and δ_2 is computed from $(x'_1, x'_2, \dots, x'_k)$ and $(x'_{k+t+1}, x'_{k+t+2}, \dots, x'_{k+r})$. Each of these syndrome computations involves $(k + t)$ residue digits. Since the dynamic ranges of δ_1 and δ_2 are much smaller than M_N , each syndrome value may map to more than one error vector of no more than t residue digit errors of different locations l_i and magnitudes e_{l_i} . Consequently, two syndromes are insufficient to correct the error(s) introduced into the three distinct groups of residue digits. A third syndrome δ_3 computed from $(x'_{k+1}, x'_{k+2}, \dots, x'_{k+t})$ and $(x'_{k+t+1}, x'_{k+t+2}, \dots, x'_{k+r})$ is needed. By finding the common error vector represented by the three syndromes, residue digit errors can be determined exactly provided that $M_U > M_K$. If the modulo operations are further decomposed into even smaller modulo operations by using more residue groups, it may not be possible to detect up to t residue digit errors when less than t redundant residues are involved in the syndrome computations.

The three syndromes δ_1 , δ_2 and δ_3 can be computed as follows:

$$\delta_1 = \left| \tilde{X}_U - \tilde{X}_K \right|_{M_U} \quad (5.33)$$

$$\delta_2 = \left| \tilde{X}_V - \tilde{X}_K \right|_{M_V} \quad (5.34)$$

$$\delta_3 = \left| \tilde{X}_V - \tilde{X}_U \right|_{M_V} \quad (5.35)$$

By substituting Equations (5.24)-(5.26) into Equations (5.33)-(5.35),

$$\delta_1 = \left| X - X \right|_{M_U} = 0 \quad (5.36)$$

$$\delta_2 = \left\| X \right|_{M_V} - X \Big|_{M_V} = \left| - \left\lfloor \frac{X}{M_V} \right\rfloor M_V \right|_{M_V} = 0 \quad (5.37)$$

$$\delta_3 = \left\| X \right|_{M_V} - X \Big|_{M_V} = 0 \quad (5.38)$$

Therefore, when there is no erroneous residue digit, the syndromes $\delta_1 = \delta_2 = \delta_3 = 0$. If any residue digit is in error, the error location category can also be identified from the values of the three syndromes. By substituting Equations (5.27)-(5.29) into Equations (5.33)-(5.35), the values of δ_1 , δ_2 and δ_3 for each possible error location category can be computed as follows:

EL1:

$$\delta_1 = \left| X - \left(X + \left| E_K \right|_{M_K} - \alpha_K M_K \right) \right|_{M_U} = \left| \alpha_K M_K - \left| E_K \right|_{M_K} \right|_{M_U} \quad (5.39)$$

$$\delta_2 = \left| X - \left(X + \left| E_K \right|_{M_K} - \alpha_K M_K \right) \right|_{M_V} = \left| \alpha_K M_K - \left| E_K \right|_{M_K} \right|_{M_V} \quad (5.40)$$

$$\delta_3 = \left\| X \right|_{M_V} - X \Big|_{M_V} = 0 \quad (5.41)$$

EL2:

$$\delta_1 = \left| \left(X + \left| E_U \right|_{M_U} - \alpha_U M_U \right) - X \right|_{M_U} = \left| E_U \right|_{M_U} \quad (5.42)$$

$$\delta_2 = \left\| X \right|_{M_V} - X \Big|_{M_V} = 0 \quad (5.43)$$

$$\delta_3 = \left\| X \right|_{M_V} - \left(X + \left| E_U \right|_{M_U} - \alpha_U M_U \right) \Big|_{M_V} = \left| \alpha_U M_U - \left| E_U \right|_{M_U} \right|_{M_V} \quad (5.44)$$

EL3:

$$\delta_1 = |X - X|_{M_U} = 0 \quad (5.45)$$

$$\delta_2 = \left| |X|_{M_V} - \left(X + |E_V|_{M_V} - \alpha_V M_V \right) \right|_{M_V} = |E_V|_{M_V} \quad (5.46)$$

$$\delta_3 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - X \right|_{M_V} = |E_V|_{M_V} \quad (5.47)$$

EL4:

$$\delta_1 = \left| \left(X + |E_U|_{M_U} - \alpha_U M_U \right) - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_U} = \left| |E_U|_{M_U} - |E_K|_{M_K} + \alpha_K M_K \right|_{M_U} \quad (5.48)$$

$$\delta_2 = \left| |X|_{M_V} - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_V} = \left| \alpha_K M_K - |E_K|_{M_K} \right|_{M_V} \quad (5.49)$$

$$\delta_3 = \left| |X|_{M_V} - \left(X + |E_U|_{M_U} - \alpha_U M_U \right) \right|_{M_V} = \left| \alpha_U M_U - |E_U|_{M_U} \right|_{M_V} \quad (5.50)$$

EL5:

$$\delta_1 = \left| X - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_U} = \left| \alpha_K M_K - |E_K|_{M_K} \right|_{M_U} \quad (5.51)$$

$$\delta_2 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_V} = \left| |E_V|_{M_V} - |E_K|_{M_K} + \alpha_K M_K \right|_{M_V} \quad (5.52)$$

$$\delta_3 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - X \right|_{M_V} = |E_V|_{M_V} \quad (5.53)$$

EL6:

$$\delta_1 = \left| \left(X + |E_U|_{M_U} - \alpha_U M_U \right) - X \right|_{M_U} = |E_U|_{M_U} \quad (5.54)$$

$$\delta_2 = \left| |X|_{M_V} - \left(X + |E_V|_{M_V} - \alpha_V M_V \right) \right|_{M_V} = |E_V|_{M_V} \quad (5.55)$$

$$\delta_3 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - \left(X + |E_U|_{M_U} - \alpha_U M_U \right) \right|_{M_V} = \left| |E_V|_{M_V} - |E_U|_{M_U} + \alpha_U M_U \right|_{M_V} \quad (5.56)$$

EL7:

$$\delta_1 = \left| \left(X + |E_U|_{M_U} - \alpha_U M_U \right) - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_U} = \left| |E_U|_{M_U} - |E_K|_{M_K} + \alpha_K M_K \right|_{M_U} \quad (5.57)$$

$$\delta_2 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - \left(X + |E_K|_{M_K} - \alpha_K M_K \right) \right|_{M_V} = \left| |E_V|_{M_V} - |E_K|_{M_K} + \alpha_K M_K \right|_{M_V} \quad (5.58)$$

$$\delta_3 = \left| \left(|X|_{M_V} + |E_V|_{M_V} - \alpha_V M_V \right) - \left(X + |E_U|_{M_U} - \alpha_U M_U \right) \right|_{M_V} = \left| |E_V|_{M_V} - |E_U|_{M_U} + \alpha_U M_U \right|_{M_V} \quad (5.59)$$

5.3.2 Proposed Error Detection and Correction Procedure

From Equations (5.39)-(5.59), there are nine different syndrome expressions and each syndrome can assume three of these nine expressions, as shown in Table 5.2. Table 5.3 shows the mapping of the error location categories to the expressions of the three syndromes. Every syndrome value in a column should map to an error vector in the error location category of the column, and every error vector in an error location category should also be mapped to at least one of the three syndrome values in the column.

From Tables 5.2 and 5.3, six lookup tables are constructed. In each lookup table, the lookup values for the error vectors are pre-computed and stored along with their error vectors. If a search key to the lookup table matches the lookup value, the error vectors corresponding to the lookup value will be retrieved. Table 5.4 shows the types of error magnitudes (E_K , E_U , E_V) corresponding to the error vectors stored in each lookup table, the lookup values (computed by v_1 to v_9) and the search key (δ_1 , δ_2 or δ_3) used for the identification of error location categories (**EL1** to **EL7**) and the retrieval of error vectors. For example, from the first row of Table 5.4, it can be deduced that T_1 is constructed for the retrieval of the error vector of E_K for **EL1**, **EL4** and **EL5**. The lookup values, v_2 and v_4 , of T_1 are computed for all possible combinations of $\alpha_K \in \{0, 1\}$ and E_K of t or fewer residue digit errors. If the error(s) is(are) identified to be in **EL1** or **EL5** (or **EL1** or **EL4**), its error vector of E_K can be retrieved from T_1 by the syndrome δ_1 (or δ_2). The exact error vector is recovered in two stages. The error location category is to be identified before the corresponding error vector retrieved from the lookup table is used to correct the erroneous residue digits.

Table 5.2 Possible Syndrome Expressions for Error Location Categories **EL1– EL7**.

δ	Matched values	Error vectors			Categories
		E_K	E_U	E_V	
δ_1	$v_1 = E_U _{M_U} - E_K _{M_K} + \alpha_K M_K _{M_U}$	✓	✓		EL4, EL7
	$v_2 = \alpha_K M_K - E_K _{M_K} _{M_U}$	✓			EL1, EL5
	$v_3 = E_U _{M_U}$		✓		EL2, EL6
δ_2	$v_4 = \alpha_K M_K - E_K _{M_K} _{M_V}$	✓			EL1, EL4
	$v_5 = E_V _{M_V} - E_K _{M_K} + \alpha_K M_K _{M_V}$	✓		✓	EL5, EL7
	$v_6 = E_V _{M_V}$			✓	EL3, EL6
δ_3	$v_7 = \alpha_U M_U - E_U _{M_U} _{M_V}$		✓		EL2, EL4
	$v_8 = E_V _{M_V}$			✓	EL3, EL5
	$v_9 = E_V _{M_V} - E_U _{M_U} + \alpha_U M_U _{M_V}$		✓	✓	EL6, EL7

Table 5.3 Mapping of Error Location Categories to Syndromes.

	EL1	EL2	EL3	EL4	EL5	EL6	EL7
δ_1	v_2	v_3	0	v_1	v_2	v_3	v_1
δ_2	v_4	0	v_6	v_4	v_5	v_6	v_5
δ_3	0	v_7	v_8	v_7	v_8	v_9	v_9

If only one of the three syndromes is zero, as indicated in Table 5.3, the error location category can be identified by the zero syndrome. It is **EL1** for $\delta_3 = 0$, **EL2** for $\delta_2 = 0$ and **EL3** for $\delta_1 = 0$. The error vectors for **EL1**, **EL2** and **EL3** can be retrieved from the lookup tables T_1 , T_2 and T_3 , respectively by using any of the two non-zero syndromes. If none of the syndromes is zero, the error location category could be either **EL4**, **EL5**, **EL6** or **EL7** as indicated in Table 5.3. In this case, the actual error location category can be identified by checking if the error vectors retrieved by δ_1 , δ_2 and δ_3 are consistent according to Table 5.4. For example, since $E_{KV} = E_K \cup E_V$, if the errors fall in **EL5**, the error vector of E_{KV} retrieved from T_5 by δ_2 should subsume the error vectors of E_K and E_V , retrieved from T_1 and T_3 by δ_1 and δ_3 , respectively. Upon locating the actual error location category, the erroneous residue digit(s) is(are) corrected by subtracting the retrieved error vector from the received residue digits.

Let $E = \mathbf{find}(T, \delta)$ be the table lookup function that returns an error vector of E for the search key δ to the lookup table T or null if no matching record is found. The steps required for multiple residue digit error detection and correction are:

1. Decode $\tilde{X}_K$, $\tilde{X}_U$ and $\tilde{X}_V$ from the received residue digits.
2. Compute δ_1 , δ_2 and δ_3 by Equations (5.33), (5.34) and (5.35), respectively.
3. If $\delta_1 = \delta_2 = \delta_3 = 0$, there is no error. Goto Step 10.
4. If only one of δ_1 , δ_2 and δ_3 is non-zero, declare that there are more than t residue digit errors. Goto Step 10.
5. If $\delta_1 \neq 0$, $\delta_2 \neq 0$ and $\delta_3 = 0$, the erroneous residue digit(s) is(are) in **EL1**. $E_{EL1} = \mathbf{find}(T_1, \delta_2)$. Subtract the error vector of E_{EL1} from the received residue digit(s) to correct the erroneous residue digit(s). Goto Step 10.
6. If $\delta_1 \neq 0$, $\delta_2 = 0$ and $\delta_3 \neq 0$, the erroneous residue digit(s) is(are) in **EL2**. $E_{EL2} = \mathbf{find}(T_2, \delta_1)$. Subtract the error vector of E_{EL2} from the received residue digit(s) to correct the erroneous residue digit(s). Goto Step 10.
7. If $\delta_1 = 0$, $\delta_2 \neq 0$ and $\delta_3 \neq 0$, the erroneous residue digit(s) is(are) in **EL3**. $E_{EL3} = \mathbf{find}(T_3, \delta_2 \text{ or } \delta_3)$. Subtract the error vector of E_{EL3} from the received residue digit(s) to correct the erroneous residue digit(s). Goto Step 10.
8. If $\delta_1 \neq 0$, $\delta_2 \neq 0$ and $\delta_3 \neq 0$, the erroneous residue digit(s) could be in **EL4**, **EL5**, **EL6** or **EL7**. The exact error location category is identified as follows:
9. If $E_{EL4} = \mathbf{find}(T_4, \delta_1) = \mathbf{find}(T_1, \delta_2) \cup \mathbf{find}(T_2, \delta_3) \neq 0$, the erroneous residue digit(s) is(are) in **EL4**. Set $E = E_{EL4}$.
10. If $E_{EL5} = \mathbf{find}(T_5, \delta_2) = \mathbf{find}(T_1, \delta_1) \cup \mathbf{find}(T_3, \delta_3) \neq 0$, the erroneous residue digit(s) is(are) in **EL5**. Set $E = E_{EL5}$.
11. If $E_{EL6} = \mathbf{find}(T_6, \delta_3) = \mathbf{find}(T_2, \delta_1) \cup \mathbf{find}(T_3, \delta_2) \neq 0$, the erroneous residue digit(s) is(are) in **EL6**. Set $E = E_{EL6}$.
12. If $\mathbf{find}(T_4, \delta_1) \cap \mathbf{find}(T_5, \delta_2) \neq 0$ or $\mathbf{find}(T_4, \delta_1) \cap \mathbf{find}(T_6, \delta_3) \neq 0$ or $\mathbf{find}(T_5, \delta_2) \cap \mathbf{find}(T_6, \delta_3) \neq 0$, the erroneous residue digit(s) is(are) in **EL7**. Set $E = E_{EL7} = \mathbf{find}(T_4, \delta_1) \cup \mathbf{find}(T_5, \delta_2)$.

13. If there is none or more than one error location category found, declare that there are more than t residue digit errors. Goto Step 10. Otherwise, subtract the error vector of E from the received residue digit(s) to correct the erroneous residue digit(s).
14. End procedure.

In Step 8, each lookup table is accessed by at most three different syndromes. Different lookup tables can be accessed simultaneously by the same syndrome but the same lookup table cannot be simultaneously accessed by different syndromes. Thus, the retrieval of error vector can be completed in no more than three lookup cycles. As the search keys to these lookup tables are the three syndromes of sizes M_U and M_V , the sizes and access speed of the lookup tables become dependent on the number and sizes of the moduli of RRNS. This problem can be mitigated by using multi-level lookup tables with smaller address input bit widths and less complicated address decoding circuits.

The flow chart of the above procedure is depicted in Figure 5.2. It should be noted that **EL7** is possible only if there are at least three residue digit errors. Therefore, for double residue digit error detection and correction, the step to identify **EL7** can be omitted.

Table 5.4 Characteristics of Lookup Tables for Error Vector Retrieval.

Table	Error vectors			lookup value	Search key	Error location groups
	E_K	E_U	E_V			
T_1	✓			v_2	δ_1	EL1, EL5
				v_4	δ_2	EL1, EL4
T_2		✓		v_3	δ_1	EL2, EL6
				v_7	δ_3	EL2, EL4
T_3			✓	v_6	δ_2	EL3, EL6
				v_8	δ_3	EL3, EL5
T_4	✓	✓		v_1	δ_1	EL4, EL7
T_5	✓		✓	v_5	δ_2	EL5, EL7
T_6		✓	✓	v_9	δ_3	EL6, EL7

Figure 5.2 Proposed multiple residue digit error detection and correction procedure.

5.3.3 Illustrative Example

Consider the moduli set $\{7, 9, 11, 13, 17, 19, 23, 25, 29\}$ where $\{7, 9, 11\}$ are the information moduli and $\{13, 17, 19, 23, 25, 29\}$ are the redundant moduli. Thus, $M_K = 693$, $M_U = 4,199$ and $M_V = 16,675$. Since the number of redundant moduli is 6, this RRNS is capable of correcting three residue digit errors. Let the error free residue representation be $X = 246 \equiv (1, 3, 4, 12, 8, 18, 16, 21, 14)$. Assume that three residue digit errors are introduced into X and the error vector is $(0, 2, 0, 0, 0, 6, 7, 0, 0)$. $E_l = 2,307,079,775$. $\tilde{X} = |X + E_l|_{M_N} = 2,307,080,021 \equiv (1, 5, 4, 12, 8, 5, 0, 21, 14)$. Therefore, $\tilde{X}_K \equiv (1, 5, 4) = 554$, $\tilde{X}_U \equiv (12, 8, 5) = 2,456$ and $\tilde{X}_V \equiv (0, 21, 14) =$

10,396. Using Equations (5.34)-(5.36), the three syndromes are computed as $\delta_1 = |2,456-554|_{4,199} = 1,902$, $\delta_2 = |10,396-554|_{16,675} = 9,842$ and $\delta_3 = |10,396-2,456|_{16,675} = 7,940$. Since $\delta_1 \neq 0$, $\delta_2 \neq 0$ and $\delta_3 \neq 0$, the error location category is either **EL4**, **EL5**, **EL6** or **EL7**.

From the error vector of E_i , $E_K \equiv (0, 2, 0, 0, 0, 0, 0, 0, 0) = 26,957,055,125$, $E_U \equiv (0, 0, 0, 0, 0, 0, 6, 0, 0, 0) = 28,092,089,025$ and $E_V \equiv (0, 0, 0, 0, 0, 0, 7, 0, 0, 0) = 44,303,334,075$. Hence, $|E_K|_{M_K} = 308$, $|E_U|_{M_U} = 2,210$ and $|E_V|_{M_V} = 10,150$. With reference to Table 5.2, the possible syndrome values are: $v_1 = |2,210-308|_{4,199} = 1,902$ for $\alpha_K = 0$ and $|19,02+693|_{4,199} = 2,595$ for $\alpha_K = 1$; $v_2 = |-308|_{4,199} = 3,891$ for $\alpha_K = 0$ and $|693-308|_{4,199} = 385$ for $\alpha_K = 1$; $v_3 = 2,210$; $v_4 = |-308|_{16,675} = 16,367$ for $\alpha_K = 0$ and $|693-308|_{16,675} = 385$ for $\alpha_K = 1$; $v_5 = |10,150-308|_{16,675} = 9,842$ for $\alpha_K = 0$ and $|9,842+693|_{16,675} = 10,535$ for $\alpha_K = 1$; $v_6 = 10,150$; $v_7 = |-2,210|_{16,675} = 14,465$ for $\alpha_U = 0$ and $|4,199-2,210|_{16,675} = 1,989$ for $\alpha_U = 1$; $v_8=10,150$; $v_9 = |10,150-2,210|_{16,675} = 7,940$ for $\alpha_U = 0$ and $|10,150-2,210+4,199|_{16,675} = 12,139$ for $\alpha_U = 1$.

These values correspond to **EL4-EL7** are listed in Table 5.5.

From Table 5.5, it is observed that the magnitudes of $\delta_1 = 1,902$, $\delta_2 = 9,842$ and $\delta_3 = 7,940$ computed directly from the received residue representation by Equations (5.33)-(5.35) match those computed by using Table 5.2 for **EL7** in the lookup tables. The error location category is therefore **EL7**. Since $\mathbf{find}(T_4, \delta_1 = 1,902) = (0, 2, 0, 0, 0, 6, 0, 0, 0, 0)$ and $\mathbf{find}(T_5, \delta_2 = 9,842) = (0, 2, 0, 0, 0, 0, 7, 0, 0, 0)$, $\mathbf{find}(T_4, \delta_1 = 1,902) \cap \mathbf{find}(T_5, \delta_2 = 9842) = (0, 2, 0, 0, 0, 0, 0, 0, 0, 0) \neq 0$. Therefore, the error vector is $E = E_{EL7} \equiv (0, 2, 0, 0, 0, 6, 0, 0, 0, 0) \cup (0, 2, 0, 0, 0, 0, 7, 0, 0, 0) = (0, 2, 0, 0, 0, 6, 7, 0, 0, 0)$ and $\tilde{X} - E = (1, 5, 4, 12, 8, 5, 0, 21, 14) - (0, 2, 0, 0, 0, 6, 7, 0, 0, 0) = (1, 3, 4, 12, 8, -1, -7, 21, 14) = (1, 3, 4, 12, 8, 18, 16, 21, 14) \equiv X$.

Table 5.5 Syndrome Values for **EL4-EL7**.

	EL4	EL5	EL6	EL7
δ_1	1,902 or 2595	3,891 or 385	2210	1,902 or 2,595
δ_2	16,387 or 385	9,842 or 10,535	10150	9,842 or 10,535
δ_3	14,465 or 1,989	10,150	7,940 or 12,139	7,940 or 12,139

5.3.4 Proof of Uniqueness

The proposed procedure assumes that each syndrome value represents some combinations of residue digit errors in the respective group of residue digits, $\tilde{X}_K$, $\tilde{X}_U$ or $\tilde{X}_V$, such that the error vectors retrieved from the lookup tables, T_1 , T_2 and T_3 , by one of the two nonzero syndromes in Steps 5 to 7 are unique and unambiguous when there are t or fewer residue digit errors. Since the calculations of the three syndromes in Equations (5.33)-(5.35) involve either modulo M_U or modulo M_V operation, δ_1 , δ_2 and δ_3 are bounded by $[0, M_U-1]$, $[0, M_V-1]$ and $[0, M_V-1]$, respectively. To ensure the uniqueness of the syndrome representation, the redundant moduli must be chosen such that M_U and M_V are larger than the number of possible error vectors.

The error digit e_i of a modulo channel m_i is bounded by $[1, m_i-1]$. Therefore, the number of different combinations of single residue digit error in modulo channel m_i is (m_i-1) . Similarly, the number of error vectors due to multiple residue digit errors can be computed by the product of the number of error vectors due to single digit error in each of the erroneous modulo channels.

The number of error vectors due to a single residue digit error in q moduli is given by:

$$S_1 = \sum_{l_1=1}^q (m_{l_1} - 1) = \sum_{l_1=1}^q m_{l_1} - C_1^q \quad (5.60)$$

The number of error vectors due to double residue digit errors in q moduli is

$$\begin{aligned} S_2 &= \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q (m_{l_2} - 1)(m_{l_1} - 1) \\ &= \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q (m_{l_2} m_{l_1} - m_{l_2} - m_{l_1} + 1) \\ &= \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q m_{l_2} m_{l_1} - \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q (m_{l_2} + m_{l_1}) + C_2^q \\ &= \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q m_{l_2} m_{l_1} - C_2^{q-1} \sum_{l_1=1}^q m_{l_1} + C_2^q \end{aligned} \quad (5.61)$$

The number of error vectors due to triple residue digit errors in q moduli is

$$\begin{aligned}
S_3 &= \sum_{l_3=1}^{q-2} \sum_{l_2=l_3+1}^{q-1} \sum_{l_1=l_2+1}^q (m_{l_3}-1)(m_{l_2}-1)(m_{l_1}-1) \\
&= \sum_{l_3=1}^{q-2} \sum_{l_2=l_3+1}^{q-1} \sum_{l_1=l_2+1}^q m_{l_3} m_{l_2} m_{l_1} - C_1^{q-2} \sum_{l_2=1}^{q-1} \sum_{l_1=l_2+1}^q m_{l_2} m_{l_1} \\
&\quad + C_2^{q-1} \sum_{l_1=1}^q m_{l_1} - C_3^q
\end{aligned} \tag{5.62}$$

In general, the number of error vectors due to t residue digit errors in q moduli is

$$\begin{aligned}
S_t &= \sum_{l_t=1}^{q-t+1} \dots \sum_{l_1=l_t+1}^q (m_{l_t} m_{l_{t-1}} \dots m_{l_1}) + \dots \\
&\quad + (-1)^{t-1} C_{t-1}^{q-1} \sum_{l_1=1}^q m_{l_1} + (-1)^t C_t^q
\end{aligned} \tag{5.63}$$

The total number of error vectors due to t or less residue digit errors in q moduli is

$$S_{q,t} = \sum_{i=1}^t S_i \tag{5.64}$$

When the error location category is **EL1**, the total number of error vectors due to t or less residue digit errors in k moduli of $\tilde{X}_K$ can be computed by Equation (5.64) as follows:

$$\begin{aligned}
S_{EL1} &= S_{k,t} = \sum_{i=1}^t S_i \\
&= \sum_{l_1=1}^k (m_{l_1}-1) + \sum_{l_2=1}^{k-1} \sum_{l_1=l_2+1}^k (m_{l_2}-1)(m_{l_1}-1) + \dots + \sum_{l_t=1}^{k-t+1} \dots \sum_{l_1=l_t+1}^k (m_{l_t}-1) \dots (m_{l_1}-1)
\end{aligned} \tag{5.65}$$

From Equation (5.40), the syndrome values in the lookup table T_1 that map to all these S_{EL1} number of error vectors are computed by $\left| \alpha_K M_K - |E_K|_{M_K} \right|_{M_V}$, where $\alpha_K \in \{0,1\}$. Due to the offset of M_K , M_V must be more than twice the number of possible error vectors to ensure that every error vector due to the residue digit errors in **EL1** can be uniquely retrieved from T_1 by δ_2 , i.e.,

$$M_V > 2 \times S_{EL1} \tag{5.66}$$

When the error location category is **EL2**, the total number of combinations of t or less residue digit errors in t moduli of $\tilde{X}_U$ can be computed by Equation (5.64) as follows:

$$S_{EL2} = S_{t,t} = \sum_{l_1=1}^t (m_{l_1} - 1) + \sum_{l_2=1}^{t-1} \sum_{l_1=l_2+1}^t (m_{l_2} - 1)(m_{l_1} - 1) + \dots + \sum_{l_t=1}^1 \dots \sum_{l_1=t}^t (m_{l_t} - 1) \dots (m_{l_1} - 1) \quad (5.67)$$

Upon simplification, Equation (5.67) becomes

$$S_{t,t} = (m_{l_t} m_{l_{t-1}} \dots m_{l_1}) - 1 = \prod_{i=l_1}^{l_t} m_i - 1 \quad (5.68)$$

Since the index i of m_i ranges from $k+1$ to $k+t$ for the moduli in $\tilde{X}_U$, Equation (5.68) can be expressed as:

$$S_{EL2} = S_{t,t} = \prod_{i=k+1}^{k+t} m_i - 1 = M_U - 1 \quad (5.69)$$

From Equation (5.42), the syndrome values in the lookup table T_2 that map to all these S_{EL2} error vectors are computed by $|E_U|_{M_U}$. To ensure that every possible error vector in **EL2** can be uniquely retrieved by δ_1 , $M_U > S_{EL2}$, which is always satisfied.

For error location category **EL3**, the total number of error vectors due to t or less residue digit errors in the t moduli of $\tilde{X}_V$ can be computed using Equation (5.68) as:

$$S_{EL3} = \prod_{i=k+t+1}^{k+2t} m_i - 1 = M_V - 1 \quad (5.70)$$

These S_{EL3} number of error vectors from the lookup table T_3 can be retrieved by $\delta_2 = |E_V|_{M_V}$ from Equations (5.46) and (5.47). Following the same argument, $M_V > S_{EL3}$, which is also always satisfied.

When all three syndromes are non-zero, the error location category could be **EL4**, **EL5**, **EL6** or **EL7**. To avoid ambiguity, only one unique error vector of one error location category can be retrieved by the set of three syndromes. Let δ_{l_1} , δ_{l_2} and δ_{l_3} be the syndromes used to identify an

error vector in one error location category and δ_{1_j} , δ_{2_j} and δ_{3_j} be another set of syndromes used to identify an error vector in another category among **EL4**, **EL5**, **EL6** and **EL7**. Due to the modulo operations, ambiguity in error detection can arise when

$$\delta_{1_i} - \delta_{1_j} = 0, \quad (5.71)$$

$$\delta_{2_i} - \delta_{2_j} = 0, \quad (5.72)$$

$$\text{and } \delta_{3_i} - \delta_{3_j} = 0. \quad (5.73)$$

From Table 5.3, collision occurs between **EL4** and **EL5** if and only if $v_1 - v_2 = 0$, $v_4 - v_5 = 0$ and $v_7 - v_6 = 0$. By substituting Equations (5.48) and (5.51) into Equations (5.71), Equations (5.49) and (5.52) into Equation (5.72), and Equations (5.50) and (5.53) into Equation (5.73),

$$\left| E_{U_i} \right|_{M_U} - \left| E_{K_i} \right|_{M_K} + \left| E_{K_j} \right|_{M_K} + (\alpha_{K_i} - \alpha_{K_j}) M_K = \beta_1 M_U \quad (5.74)$$

$$-\left| E_{K_i} \right|_{M_K} - \left| E_{V_j} \right|_{M_V} + \left| E_{K_j} \right|_{M_K} + (\alpha_{K_i} - \alpha_{K_j}) M_K = \beta_2 M_V \quad (5.75)$$

$$\alpha_{U_i} M_U - \left| E_{U_i} \right|_{M_U} - \left| E_{V_j} \right|_{M_V} = \beta_3 M_V \quad (5.76)$$

where

$$\beta_1 = \left\lfloor \frac{\left| E_{U_i} \right|_{M_U} - \left| E_{K_i} \right|_{M_K} + \alpha_{K_i} M_K}{M_U} \right\rfloor - \left\lfloor \frac{\alpha_{K_j} M_K - \left| E_{K_j} \right|_{M_K}}{M_U} \right\rfloor \quad (5.77)$$

$$\beta_2 = \left\lfloor \frac{\alpha_{K_i} M_K - \left| E_{K_i} \right|_{M_K}}{M_V} \right\rfloor - \left\lfloor \frac{\left| E_{V_j} \right|_{M_V} - \left| E_{K_j} \right|_{M_K} + \alpha_{K_j} M_K}{M_V} \right\rfloor \quad (5.78)$$

$$\beta_3 = \left\lfloor \frac{\alpha_{U_i} M_U - \left| E_{U_i} \right|_{M_U}}{M_V} \right\rfloor \quad (5.79)$$

Subtracting Equations (5.74) and (5.76) from Equation (5.75),

$$\begin{aligned} -\alpha_{U_i} M_U &= \beta_2 M_V - \beta_1 M_U - \beta_3 M_V \\ (\beta_1 - \alpha_{U_i}) M_U &= (\beta_2 - \beta_3) M_V \\ \frac{M_U}{M_V} &= \frac{\beta_2 - \beta_3}{\beta_1 - \alpha_{U_i}} \end{aligned} \quad (5.80)$$

Since M_V and M_U are relatively prime, Equation (5.80) is correct iff $M_U = \beta_2 - \beta_3$ and $M_V = \beta_1 - \alpha_{U_i}$. The validity of (5.25) in Section 5.3.1 requires that $M_U > M_K$. Thus, β_1 in Equation (5.77) is an integer in the range $[0, 1]$. Since $\alpha_{U_i} \in \{0, 1\}$, the range of $\beta_1 - \alpha_{U_i}$ is $[-1, 1]$. M_V is the product of t redundant moduli and it will always be larger than 2. Since $M_V \neq \beta_1 - \alpha_{U_i}$, Equation (5.80) is invalid, the set of syndromes used for retrieving the error vector in **EL4** will never collide with that in **EL5**. Similar proof can be obtained to validate that there is no collision between the two sets of syndromes in any other pairs of error location categories.

In summary, the proposed procedure is able to identify and correct up to t erroneous residue digits provided that the following two conditions are fulfilled.

$$\text{Condition 1: } M_V > 2 \times S_{EL1};$$

$$\text{Condition 2: } M_U > M_K.$$

For the example in Section 5.3.3, since $M_U = 4199$ is larger than $M_K = 693$, *Condition 2* is fulfilled. Using Equation (5.65), S_{EL1} is computed to be 692. *Condition 1* is also fulfilled because $M_V > 2 \times S_{EL1} = 1384$. It is also observed in Table 5.5 that the conditions for collision, Equations (5.71)-(5.73), are not met by the sets of syndrome values between any two columns.

For highly fault-tolerant computing system where $t = k$, one easy way to fulfil *Condition 1* and *2* is by choosing the moduli from prime numbers with comparable word lengths. Otherwise, when $t < k$, the redundant moduli can be chosen from an additional pool of coprime numbers in the forms of 2^n and $2^n \pm 1$ that are of larger word lengths than the information moduli such that *Conditions 1* and *2* are always fulfilled. Since modulo 2^n and $2^n \pm 1$ arithmetic operations are more hardware efficient, the differences in latency between the information and redundant modulus channels will not be significant.

5.3.5 Error Decoding for Reduced Redundant Space

The algorithm proposed in Section 5.3.2 can detect and correct multiple residue errors with a fixed number of syndrome computations regardless of the size of moduli set. The cost for the deterministic bound in computation time is the requirement of larger redundant space due to *Condition 2*. This condition can be completely eliminated by increasing the number of syndrome computations and lookup tables according to the number of information moduli.

To eliminate *Condition 2*, the received information residues are divided into $\lceil k/t \rceil$ groups of t residues each except the last group, which may have less than t residues, i.e.,

$$\tilde{X}_K^{(1)} \equiv (\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_t) \quad , \quad \tilde{X}_K^{(2)} \equiv (\tilde{x}_{t+1}, \tilde{x}_{t+2}, \dots, \tilde{x}_{2t}) \quad , \quad \dots \quad , \quad \tilde{X}_K^{(\lceil k/t \rceil)} \equiv (\tilde{x}_{(\lceil k/t \rceil-1)t+1}, \tilde{x}_{(\lceil k/t \rceil-1)t+2}, \dots, \tilde{x}_k) \quad .$$

Including the two redundant residue groups, $\tilde{X}_U$ and $\tilde{X}_V$, there are altogether $\lceil k/t \rceil + 2$ groups of residues. By excluding one of the $\lceil k/t \rceil + 2$ residue groups in each syndrome computation, $\lceil k/t \rceil + 2$ syndromes needed be computed and any errors occurred in each group of residue digits can be detected by checking if one of the computed syndromes is zero.

Let $\tilde{X}_{UK,(i)} \equiv (\tilde{X}_U, \tilde{X}_K^{(1)}, \dots, \tilde{X}_K^{(i-1)}, \tilde{X}_K^{(i+1)}, \dots, \tilde{X}_K^{(\lceil k/t \rceil)})$ denote the magnitude due to all residues but

$$\tilde{X}_K^{(i)} \text{ in } \tilde{X}_U \text{ and } \tilde{X}_K \text{ . In this case, } M_K = \prod_{i=1}^k m_i \text{ , } M_K^{(i)} = \prod_{j=1}^t m_{(i-1)+j} \text{ and } M_{UK,(i)} = \frac{M_U \times M_K}{M_K^{(i)}} \text{ .}$$

Also, let $E_K^{(i)} \equiv (e_{k_1}, e_{k_2}, \dots, e_{k_j}, 0, \dots, 0)$ denote the magnitude of the error vector introduced into

$\tilde{X}_K^{(i)}$, where $k_1, k_2, \dots, k_j$ for $j \leq t$ represent the positions of the error digits. Similarly,

$E_{UK,(i)} \equiv (e_{k_1}, \dots, e_{k_j}, e_{u_1}, \dots, e_{u_l}, 0, \dots, 0)$ denotes the magnitude of the error vector introduced

into $\tilde{X}_{UK,(i)}$, where $k_1, k_2, \dots, k_j$ and $u_1, u_2, \dots, u_l$ represent the positions of the error digits and

$j+l \leq t$.

$\tilde{X}_{UK,(i)}$ can be expressed in term of $\tilde{X}$ as:

$$\tilde{X}_{UK,(i)} = \left| \tilde{X} \right|_{M_{UK,(i)}} \quad (5.81)$$

When there is no error, $\tilde{X} = X$ and

$$\tilde{X}_{UK,(i)} = \left| X \right|_{M_{UK,(i)}} \quad (5.82)$$

Since $M_{UK,(i)} > M_K$, Equation (5.82) can be rewritten as:

$$\tilde{X}_{UK,(i)} = X \quad (5.83)$$

When the number of erroneous residues is less than or equal to t ,

$$\begin{aligned} \tilde{X}_{UK,(i)} &= \left| X + E_{UK,(i)} \right|_{M_{UK,(i)}} \\ &= X + \left| E_{UK,(i)} \right|_{M_{UK,(i)}} - \alpha_{M_{UK,(i)}} M_{UK,(i)} \end{aligned} \quad (5.84)$$

$$\text{where } \alpha_{M_{UK,(i)}} = \begin{cases} 0 & \text{if } X + \left| E_{UK,(i)} \right|_{M_{UK,(i)}} < M_{UK,(i)} \\ 1 & \text{if } X + \left| E_{UK,(i)} \right|_{M_{UK,(i)}} \geq M_{UK,(i)} \end{cases}.$$

Similar approach to the detection and identification of erroneous residue digit locations can be performed by checking the error vectors retrieved from the lookup tables using the following $\lceil k/t \rceil + 2$ syndromes as search keys to the lookup tables.

$$\delta_1 = \left| \tilde{X}_U - \tilde{X}_K \right|_{M_U} \quad (5.85)$$

$$\delta_2 = \left| \tilde{X}_V - \tilde{X}_K \right|_{M_V} \quad (5.86)$$

$$\delta_3 = \left| \tilde{X}_V - \tilde{X}_{UK,(1)} \right|_{M_V} \quad (5.87)$$

$\vdots$

$$\delta_{\lceil k/t \rceil + 2} = \left| \tilde{X}_V - \tilde{X}_{UK,(\lceil k/t \rceil)} \right|_{M_V} \quad (5.88)$$

Based on similar derivations in Equations (5.36)-(5.59) for δ_1 and δ_2 , δ_3 to $\delta_{\lceil k/t \rceil + 2}$ are the extracted error magnitudes due to the error vectors introduced into the corresponding residue groups involved in the computations.

The procedure for detecting and correcting multiple residue digits is given as follows:

1. Decode $\tilde{X}_U$, $\tilde{X}_V$ and $\tilde{X}_{UK(i)}$ for $i = [1, \lceil k/t \rceil]$ from the received residue digits.
2. Compute δ_i for $i = [1, \lceil k/t \rceil + 2]$ by Equatons (5.85) to (5.88).
3. If $\delta_1 = \delta_2 = \dots = \delta_{\lceil k/t \rceil + 2} = 0$, there is no residue error. Goto Step 8.
4. If only one of the syndromes is zero, the erroneous residue digit(s) is(are) located in one of the residue groups, $\tilde{X}_U$, $\tilde{X}_V$ or $\tilde{X}_K^{(i)}$ for $i = [1, \lceil k/t \rceil]$. Retrieve the error vector from the corresponding lookup table using one of the non-zero syndromes as the search key.
5. If more than one or less than $\lceil k/t \rceil + 2$ syndromes are zeros, declare that there are more than t erroneous residue digits. Goto Step 8.
6. If all the syndromes are non-zero, the erroneous residue digits are located in more than one residue groups. Retrieve the error vectors from the lookup tables by using δ_i for $i = [1, \lceil k/t \rceil + 2]$ as the search keys. The exact error locations are identified by checking the interception of the retrieved error vectors.
7. Subtract the error vector from the received residue digit(s) to correct the erroneous residue digit(s).
8. End procedure.

5.3.6 Comparison and Discussions

In this section, different algorithms for multiple residue digit error detection and correction algorithms are compared. Table 5.6 shows the characteristics of the proposed algorithm and [Yau73], [Sun92], [Man76], [Gol00], [Goh08]. The features compared include the number of correctable residue digit errors, the methods used for detecting and locating errors, fixed (i.e., moduli set independent) or variable (i.e., moduli set dependent) number of computation cycles, and the output domain. When there are more than one algorithm in a paper, they are identified by the suffix ‘#’ followed by a number in Table 5.6.

Among these algorithms, [Yau73], [Sun92]#3, [Man76], [Goh08]#1, [Goh08]#2 and the proposed algorithms can correct up to t residue digit errors. However, [Yau73] and [Sun92]#3 can only correct a single burst of errors. They can correct up to a maximum of t residue digit errors only in the condition that these residue digit errors occur in adjacent received residue digits. Random residue digit errors are not detectable and correctable by using these two algorithms. Algorithms [Sun92]#1 and [Sun92]#2 can detect multiple residue digit errors but the error correction capability is limited to only one or two residue digit errors, which are not particularly useful for highly fault-tolerant applications. The number of correctable residue digit errors of [Gol00]#1 is dependent on the choice of the smallest and largest moduli, m_1 and m_n . Careful selection of moduli set is required to meet the error correction specifications of an application. The number of correctable residue digit errors of [Gol00]#2 is the highest among all algorithms. Theoretically, it may correct more than t residue digit errors even with $2t$ redundant moduli.

The error detection method determines the time consumed to check if a received residue representation is erroneous. If there is no error, the procedure of locating the residue digit errors is not required. In general, the error can be detected by using either the syndrome, the CRT or the MRC method. For algorithms that use the CRT and the MRC, the N residue digits are fully decoded to determine if the magnitude of the received information falls in the illegitimate range. Decoding using the CRT involves complex and time-consuming large modulo operations. Although the modulo operations involved in the MRC are smaller, these operations have to be performed sequentially, which result in longer latency than the CRT. The syndrome computation methods of [Yau73], [Sun92]#1, [Sun92]#2 and [Sun92]#3 all require base extension and modulo subtractions. The moduli of the modulo subtractions are m_i , where $i = 1, 2, \dots, n$, which are significantly smaller than those of CRT. Moreover, only k residue digits are used in the base extension operation, which make the delay shorter than decoding n residue digits by the CRT or the MRC. The proposed algorithm also utilizes syndromes for the residue digit error detection with an entirely different method of computation. Only three (or $\lceil k/t \rceil + 2$ for the reduced redundant space version) as opposed to N modulo operations are required. The

size of each modulo operator is M_U or M_V , which is smaller than that of the CRT method but bigger than that of other syndrome computation methods. Nevertheless, the modulus of each modulo operation can be reduced by the axiom $\|X\|_a|_b = \|X\|_b$ if b divides a . By taking the modulo m_i on Equations (5.33)–(5.35) and Equations (5.85)–(5.88), where $i = k + 1, k + 2, \dots, k + r$, the moduli of the modulo operations of syndrome computations are reduced from M_U and M_V to m_i , which are comparable to the moduli of the modulo operations for other syndrome computation methods. After the modulo reduction, the proposed syndromes can be expressed as $\delta_j \equiv (\|\delta_j\|_{m_i}, \|\delta_j\|_{m_{i+1}}, \dots, \|\delta_j\|_{m_{i+t-1}})$. Among the existing methods, using syndrome to detect residue digit errors is the most efficient way because it involves smaller modulo operation and has shorter delay.

Locating the residue digit error(s) is usually more time-consuming than detecting and correcting the residue digit error(s) itself. This process dominates the overall delay of an error detection and correction algorithm. All algorithms except the proposed algorithm require at least a moderate, if not large, the number of iterative or recursive computations and comparisons to locate the error digits and the number of iterations required is also dependent on the locations of residue digit errors. The algorithm in [Yau73] groups the syndromes into a variable number of sets to locate the residue digit errors depending on the moduli set. The maximum number of sets is $\lceil (k+t-1)/(t+1) \rceil + 1$, which is the smallest of all existing algorithms. Unfortunately, such grouping of syndromes is capable of correcting only a single burst of erroneous residue digits of length less than or equal to t . The algorithms [Man76], [Gol00]#1 and [Gol00]#2 require continued fraction or integer polynomial computations to locate the residue digit error(s), which are highly iterative. The first three algorithms of [Sun92] use the consistency check method to avoid storing the error vectors. However, the arithmetic operations required by the consistency check cannot be easily implemented without using lookup tables especially for the general moduli sets. The modulus projection method of [Sun92]#4 and [Goh08]#1 use MRC and CRT, respectively to decode $n - t$ residue digits in each iteration. The decoding speed can be severely retarded by the large number of residue digits or some large modulo operations. In addition, the maximum number of iterations is C_t^N , which grows rapidly with the size of the moduli set. The

algorithm [Goh08]#2 reduces the number of residue digits involved in each iteration to r and the maximum number of iterations to $\lceil C_t^N / C_t^r \rceil$. The penalty of this solution is the binary-to-residue conversions in each iteration, which are non-trivial in general except only for the few special moduli [She04], [Low13]. The proposed algorithm uses a parallel divide-and-conquer approach to locate the residue digit errors. The received residue digits are divided into three groups and there are only seven (six for double residue digit error detection and correction) possible error location categories irrespective of the moduli set. The exact error location category can be identified and the error vector recovered simultaneously by checking the three computed syndrome values against six lookup tables in no more than three lookup cycles. The error detection tables can be implemented using pipelined multi-level lookup technique [Par97] to reduce the access time and simplify the address decoding circuit. The proposed algorithm has high parallelism and is the only algorithm that possesses a fixed and also the least number of iterations. *Condition 2* that imposes a tight constraint on the redundant space can be eliminated by allowing the number of syndromes to vary according to the number of information moduli. Even then, these syndrome computations can be computed in parallel and the identification of error locations does not involve iterative computations.

Compared with other error detection and correction codes, RRNS is unique in that it is capable of detecting and correcting arithmetic operation errors. Hence, it is beneficial to have all the operations in error detection and corrections carried out in the residue domain. Among the algorithms, the outputs of [Yau73], [Sun92]#1-#4, [Goh08]#2 and the proposed algorithms are in residue representation. All other algorithms will require additional non-trivial conversion from integer to residue representation if further arithmetic operations in RRNS are needed, which usually do.

Among the existing algorithms, only [Man76], [Goh08]#1, and [Goh08]#2 have similar error correction capabilities as the proposed algorithms. It is not possible to estimate and compare the hardware delay of the algorithm in [Man76] as it was recommended to be executed on a general purpose computer without specifying its maximum number of iterations. The maximum number of iterative computations required by [Goh08] #1 and [Goh08] #2 are specified as C_t^N and

$\lceil C_t^N / C_t^r \rceil$, respectively, but their hardware implementations are not described in [Goh08]. As their latencies are dependent on the best and worst scenarios of identifying the error locations, the following assumptions are made in order to estimate their hardware delays. (1) The algorithms are implemented based on RRNS with balanced arbitrary moduli. For simplicity, all moduli are assumed to have the same word length of w bits; (2) The arithmetic units are implemented using ROM modules. A ROM module of w -bit (address) input is assumed to have one unit of delay. (3) The delay of a ROM module is assumed to be directly proportional to the size of its input. Based on these assumptions, the delays in the three sequential stages of error detection ('Detect'), location ('Locate') and correction ('Correct'), and the overall ('Total') delays of [Goh08]#1, [Goh08]#2, the proposed algorithm ('This') and its extended version ('This (Ext)') are tabulated in Table 5.7 for different n and k . The results show that the delays of the proposed algorithms are always shorter than those of [Goh08]#1 and [Goh08]#2 under both the best ('Best') and worst ('Worst') case scenarios.

Table 5.6 Comparison of Multiple Residue Digit Error Detection and Correction Algorithms.

Algorithm	No. of errors correctable	Method		Fixed latency?	Output domain
		Detect error	Locate error		
[Yau73]	Single burst $\leq t$	Syndrome	Syndrome sets	No	Residue
[Sun92] #1	1	Syndrome	Consistency check	No	Residue
[Sun92] #2	2	Syndrome	Consistency check	No	Residue
[Sun92] #3	Single burst $\leq t$	Syndrome	Consistency check	No	Residue
[Sun92] #4	2	MRC	Modulus projection	No	Residue
[Man76]	$\leq t$	CRT	Continued fraction	No	Integer
[Gol00] #1	$\frac{(n-k)\log m_1}{\log m_1 + \log m_n}$	CRT	Unique-decode	No	Integer
[Gol00] #2	$\leq n - \sqrt{nk}$	CRT	List-decode	No	Integer
[Goh08] #1	$\leq t$	CRT	Modulus projection	No	Integer
[Goh08] #2	$\leq t$	CRT	Improved modulus projection	No	Integer & residue
This	$\leq t$	Syndrome	Syndrome check	Yes	Residue
This (Ext)	$\leq t$	Syndrome	Syndrome check	Yes	Residue

Table 5.7 Comparison of Estimated Unit Delays.

(n,k)	Algorithm	Detect	Locate		Correct	Total	
			Best	Worst		Best	Worst
(6,2)	[Goh08] #1	12	8	120	0	20	132
	[Goh08] #2	12	7	21	0	19	33
	This	8	8		1	17	
	This (Ext)	8	8		1	17	
(7,3)	[Goh08] #1	14	10	210	0	24	224
	[Goh08] #2	14	10	40	0	24	54
	This	9	8		1	18	
	This (Ext)	9	10		1	20	
(9,3)	[Goh08] #1	18	12	1008	0	30	1026
	[Goh08] #2	18	10	50	0	28	68
	This	12	12		1	25	
	This (Ext)	12	12		1	25	
(10,4)	[Goh08] #1	20	14	1680	0	34	1700
	[Goh08] #2	20	13	78	0	33	98
	This	13	12		1	26	
	This (Ext)	13	15		1	29	
(12,4)	[Goh08] #1	24	16	7920	0	40	7944
	[Goh08] #2	24	13	104	0	37	128
	This	16	16		1	33	
	This (Ext)	16	16		1	33	
(13,5)	[Goh08] #1	26	18	12870	0	44	12896
	[Goh08] #2	26	16	176	0	42	202
	This	17	16		1	34	
	This (Ext)	17	20		1	38	

5.4 Summary

The proposed single residue error detection and correction algorithm is formulated by selecting two redundant moduli fulfilling the criteria of syndrome computation. Based on the uniqueness of the syndrome, a lookup table of all possible single residue errors of an RRNS can be constructed to enable an error digit and its position to be retrieved from the syndrome of the received residues and corrected directly by a simple modular subtractor. The proposed algorithm can be implemented using an RNS reverse converter for the information residues, a simple two-residue CRT for the redundant residues, $k+3$ modular subtractors, a lookup table and simple logic gates. The proposed algorithm is less restrictive in the choice and number of moduli and more suited for hardware implementation than existing single residue digit error correction algorithms.

The proposed multiple residue error detection and correction algorithm stands out among all existing algorithms in that it requires a small and fixed number of computations and table

lookup operations regardless of the moduli set and number of correctable residue digit errors. The uniqueness criteria can be fulfilled by selecting moduli from the abundant prime numbers to meet the desired error correction capability for the application. The proposed algorithm is more efficient for hardware implementation than any other algorithms in terms of its computational latency as well as the number and sizes of its modulo operations. The moderate size modulo subtractors used for the syndrome computation can be decomposed into smaller size modulo subtractors and reused for the subtraction of the error vector from the received residue digits. The identification of error location and retrieval of error vector require only lookup tables and no arithmetic operation. To minimize the redundant space, an extended algorithm is also proposed, which has the number of syndromes and lookup tables increases with the number of information moduli, but the error locations can still be identified without requiring iterative computations.

Chapter 6

Conclusion and Future Works

6.1 Conclusions

This research aims to develop efficient algorithms and architectures to overcome the full reverse conversion into binary domain for signed integer RNS scaling and base transformation. The ambiguity in magnitude and range of the residue representation that make these operations difficult also poses significant challenges for the error decoding in RRNS. This problem is further aggravated by the use of general moduli to increase the error detection and correction capability. Hence, the time-consuming iterative computations of existing residue error detection and correction algorithms will also be addressed.

In this thesis, a new unified signed integer RNS scaler for moduli set moduli set $\{2^n-1, 2^n, 2^n+1\}$ with the scaling factor of 2^n has been presented. The proposed architecture has low complexity due to the simplification and the merger of sign detection and correction circuits. Compared to the latest published works within comparable dynamic ranges, the proposed design achieves notable improvements in terms of delay, area and power consumption as evinced by logic synthesis and post-layout physical synthesis results.

Base transformation via injective residue mapping in multi-base RNS system is a new concept introduced in this research. Instead of killing one or more moduli of a moduli set which has severe constraint on the extent of dynamic reduction, the word-lengths of the moduli are simultaneously reduced to achieve dynamic range reduction. Based on this concept, an elegant algorithm for remapping the residue digits of a signed integer from the moduli set $\{2^n-1, 2^{n+k}, 2^n+1\}$ to another set $\{2^{n-\alpha}-1, 2^{n+k-\alpha}, 2^{n-\alpha}+1\}$ of lower dynamic range has been proposed. The utilization of range quantization method eliminates unnecessary computations and its clear

advantages in hardware implementation over the best possible improvised solution have been proven using unit-gate analysis and logic synthesis results.

Finally, a new single residue error detection and correction algorithm has been proposed. By selecting two redundant moduli to fulfil the uniqueness criteria for the syndrome proposed in the algorithm, any RNS computations can be endowed with a single residue digit error correction capability. The location of the erroneous residue digit and its error magnitude can be retrieved from a lookup table by using the computed syndrome as the input. Compared to the existing algorithms, the redundant moduli selection criteria of the proposed algorithm are less restrictive. Furthermore, a non-iterative multiple residue error detection and correction algorithm has also been developed. Irrespective of the number of information and redundant moduli, only three syndromes are needed. The syndromes can be computed from three groups of received residue digits to detect and locate the erroneous residue digits. The computation time is fixed regardless of the moduli set, which makes the proposed algorithm efficient and well suited to parallel implementation on any hardware platform. The proposed algorithm has also been extended to relax the redundant moduli criteria and increase the code rate by increasing the number of syndrome computation and reducing the sizes of modular operations proportionally with the number of information moduli.

6.2 Future Works

Based on the accomplished research works presented in this thesis, the following relevant topics are identified for future explorations.

- a) Comparator is one of the vital and frequently encountered components in many DSP algorithms and applications for making branching and bounding decisions, such as image processor [She08], [Par09], test circuit [Cha04a] and general purpose processor [Suz07], [Kuc04], [Ok194]. Its efficiency can dominate algorithmic controller and data path partitioning decision in design synthesis and make a difference in the overall performance of a system. In RNS, both sign detection and residue comparison are difficult operations. By

using range detection method, these two operations can be unified to effectively and directly reduce the hardware complexity of a signed residue comparator. Meantime, range quantization approach can also be used to perform sign detection and residue comparison simultaneously. The existing residue comparators tackle only unsigned integer for magnitude comparison due to the complexities and difficulty to unify sign detection and residue comparison in RNS. The author has also developed a new algorithm based on range quantization approach for comparing signed residues representations in the four-moduli supersets, $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$ with even n . The preliminary concept and results have been presented in [Tay14a]. Basically, the dynamic range is quantized into equal subranges to facilitate fast sign detection and magnitude comparison simultaneously without the need for full magnitude recovery by the Chinese Remainder Theorem (CRT) or sequential Mixed Radix Conversion (MRC). Although the preliminary results show that these operations are less complex than those used in existing RNS magnitude comparators of comparable dynamic range, the number of terms involved in the multiplicative inverses in one of the subrange identifiers computation needs to be further reduced to improve its overall performance.

- b) The nascent concept of exploiting base transformation for dynamic range reduction sheds lights on and opens new avenues of research in the interoperability of multi-base RNS. The reverse mapping of residues from moduli set S_1 to another moduli set S_2 with larger dynamic range can be used in certain applications which require temporary expansion of dynamic range to avoid overflow. However, owing to the quantized segments of the unmapped codomain S_1 cannot be inferred directly from the domain of S_2 , the proposed range quantization method cannot be directly applied to the reverse mapping of the residues from S_2 to S_1 . New approach can be investigated to design efficient algorithm to complete the full interoperability between S_1 and S_2 without resorting to any residue-to-binary conversion.
- c) Due to the shrinking of device feature size, digital circuits become more susceptible to transient errors. For certain applications, the introduction of transient errors into the arithmetic units is intolerable. Hence, it is crucial for arithmetic units to possess real-time

error correction capability to avoid catastrophic consequences. In RRNS, the redundant residue digits always correlate with the information residue digits in arithmetic computations. This property enables the results of arithmetic operations to be easily checked after the computations and any transient error introduced into the arithmetic units can be immediately corrected. The easiest method for introducing real-time capability into arithmetic units is to attach a dedicated error checker. However, this method will incur substantial overheads. A better solution is to integrate the error checker into the arithmetic unit itself whereby the error checker shares some of the hardware with the arithmetic unit. Not only will this method hide the area and timing overheads for incorporating additional error correction capability, but it will also correct the error introduced into the error checker.

- d) For applications such as memory modules in space satellite, the rate of error occurrence differs with time and temperature. It is a waste of processing power and time if the error correctable by the fault tolerant method is higher than the worst case scenario at the specific condition. Therefore, the error correction algorithm must be adaptive to the conditions in order to provide sufficient reliability during high-upset-rate period and reduce the overheads of error checker during low-upset-rate period. The capability of error correction algorithm in RRNS is dependent on the number of redundant residue digits. The higher the number of redundant residue digits, the higher the number of correctable residue error. Thus, a reconfigurable error correction algorithm in RRNS can be designed by varying the number of redundant residue digits for error checking.

Author's Publications

Journal Papers

[Tay15a] **T. F. Tay**, C. H. Chang, and L. Sousa, “Base transformation with injective residue mapping for dynamic range reduction in RNS,” *IEEE Trans. Circuits and Syst. I*, vol. 62, no. 9, pp. 2248-2259, Sept. 2015.

[Tay15b] **T. F. Tay** and C. H. Chang, “A non-iterative multiple residue digit error detection and correction algorithm in RRNS,” *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 396-408, May. 2015.

[Tay13a] **T. F. Tay**, C. H. Chang, and J. Y. S. Low, “Efficient VLSI implementation of 2^n scaling of signed integer in RNS $\{2^n-1, 2^n, 2^n+1\}$,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 10, pp. 1936–1940, Oct. 2013.

[Cha15] C. H. Chang, Amir, A.S. Molahosseini, A.A.E. Zarandi and **T. F. Tay**, “Residue Number Systems: a new paradigm to datapath optimization for low-power and high-performance digital signal processing applications,” *IEEE Circuits and Systems Magazine*, vol. 15, no. 4, pp. 26-44, Nov. 2015.

Conference Papers

[Tay15d] **T. F. Tay** and C. H. Chang, “A new unified modular adder/subtractor for arbitrary moduli,” in *2015 Int. Symp. on Circuits and Syst.*, May 2015, Lisbon, Portugal, pp. 53-56.

[Tay14a] **T. F. Tay** and C. H. Chang, “New algorithm for signed integer comparison in four-moduli superset $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$,” in *2014 Asia Pacific Conf. Circuits and Syst.*, Nov. 2014, Okinawa, Japan, pp. 519-522.

[Tay14b] **T. F. Tay** and C. H. Chang, “A new algorithm for single residue digit error correction in Redundant Residue Number System,” in *2014 Int. Symp. on Circuits and Syst.*, May 2014, Melbourne, Australia, pp. 1748-1751.

[Low13] J. Y. S. Low, **T. F. Tay**, and C. H. Chang, “A signed integer programmable power-of-two scaler for $\{2^n-1, 2^n, 2^n+1\}$ RNS,” in *2013 Int. Symp. on Circuits and Syst.*, May 2013, Beijing, China, pp. 2211-2214.

One-year Provisional Patent

[Tay13b] **T. F. Tay** and C. H. Chang, “Efficient multiple residue error detection and correction algorithm for arithmetical processing and transmission in Residue Number System” filed on 7 Aug. 2013.

Book Chapter

[Tay16] **T. F. Tay** and C. H. Chang, “Insights into error detection and correction algorithms in Redundant Residue Number System” invited chapter in *Embedded Systems Design with Special Arithmetic and Number Systems*, L. Sousa, A. S. Molahosseini and C. H. Chang, Springer International Publishing AG, 2016 (in preparation).

References

- [Abt05] M. Abtahi and P. Siy, "The non-linear characteristic of core function of RNS numbers and its effect on RNS to binary conversion and sign detection algorithms," in *Proc. Annu. Meeting North Amer. Fuzzy Inform. Process. Soc.*, Detroit, Michigan, pp. 731-736, Jun. 2005.
- [Aka11] Y. Akasaka *et al.* (2011). *Process integration, devices, and structures (2011 ed.)*
[Online]. Available: <http://www.itrs.net/Links/2011ITRS/Home2011.htm>
- [Ali02] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology," *IEEE Trans. VLSI Syst.*, vol. 10, no. 6, pp. 806-823, Dec. 2002.
- [Avi13] S. Avik and N. Balasubramaniam, "Performance of systematic RRNS based space-time block codes with probability-aware adaptive demapping," *IEEE Trans. Wireless Commun.*, vol. 12, No. 5, pp. 2458 - 2469, May 2013.
- [Bar73] F. Barsi and P. Maestrini, "Error correcting properties of redundant residue number systems," *IEEE Trans. Comput.*, vol. C-22, pp. 307-315, Mar. 1973.
- [Bar74] F. Barsi and P. Maestrini, "Error detection and correction by product codes in residue number systems," *IEEE Trans. Comput.*, vol. C-23, pp.915-924, Sept. 1974.
- [Bar95] F. Barsi and M. C. Pinotti, "Fast base extension and precise scaling in RNS for look-up table implementations," *IEEE Trans. Signal Process.*, vol. 43, no. 10, pp. 2427-2430, Oct. 1995.
- [Bec93] P. E. Beckmann and B. R. Musicus, "Fast fault-tolerant digital convolution using a polynomial residue number system, " *IEEE Trans. Signal Process.*, vol. 41, pp. 2300-2313, July 1993.

- [Ber07] G. L. Bernocchi, G. C. Cardarilli, A. Del Re, A. Nannarelli and M. Re, "Low power adaptive filter based on RNS components," in *Proc. 2007 IEEE Int. Symp. on Circuits and Syst.*, New Orleans, USA, pp. 3211-3214, May 2007.
- [Bi08] S. Bi and W. J. Gross, "The mixed-radix Chinese Remainder Theorem and its applications to residue comparison," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1624-1632, Dec. 2008.
- [Bur93] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. VLSI Syst.*, vol. 1, no. 1, pp. 63-71, Mar. 1993.
- [Cao10] B. Cao, Y. S. Low, C. H. Chang and T. Srikanthan, "Performance analysis of different special moduli sets for RNS-based inner product step processor," in *Proc. 2010 International Conference on Green Circuits and Systems (ICGCS-2010)*, Shanghai, China, pp. 236-241, 21-23 June, 2010 (Invited paper).
- [Car04] G. C. Cardarilli, A. Del Re, A. Nannarelli and M. Re, "Low-power implementation of polyphase filters in quadratic residue number system," in *Proc. 2004 Int. Symp. on Circuits and Syst.*, Vancouver, Canada, vol. 2, pp. II-725-728, May 2004.
- [Car05] G. C. Cardarilli, A. Del Re, A. Nannarelli and M. Re, "Programmable power-of-two RNS scaler and its application to a QRNS polyphase filter," in *Proc. 2005 IEEE Int. Symp. Circuits and Syst.*, Kobe, Japan, vol. 2, pp. 1102-1105, May 2005.
- [Cha04a] A. H. Chan and G. W. Roberts, "A jitter characterization system using a component-invariant Vernier delay line," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 1, pp. 79-95, Jan. 2004.
- [Cha04b] R. Chaves and L. Sousa, " $\{2^n+1, 2^{n+k}, 2^n-1\}$: A new RNS moduli set extension," in *Proc. EUROMICRO Syst. Digit. Syst. Des. (DSD'04)*, 2004, Aug. 2004, pp. 210-217.

- [Cha11] C. H. Chang and J. Y. S. Low, "Simple, fast, and exact RNS scaler for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Trans. Circuits and Syst. I*, vol. 58, no. 11, pp. 2686-2697, Nov. 2011.
- [Che09] J. Chen and J.E. Stine, "Parallel Prefix Ling Structures for Modulo 2^n-1 Addition," *Proc. 20th IEEE Int. Conf. Application-Specific Syst., Architectures and Processors*, pp. 16-23, July 2009.
- [Che11] J. W. Chen, R. H. Yao, and W. J. Wu, "Efficient modulo 2^n+1 multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2149–2157, Dec. 2011.
- [Cho09] R Chokshi, K. S. Berezowski, A. Shrivastava, and S. J. Piestrak, "Exploiting residue number system for power-efficient digital signal processing in embedded processors," in *Proc. 2009 ACM Int. Conf. on Compilers, Architecture, and Synthesis of Embedded Systems, (CASES '09)*, New York, USA, pp. 19-28, 2009.
- [Cla93] E. D. D. Claudio, G. Orlandi, and F. Piazza, "A systolic redundant residue arithmetic error correction circuit," *IEEE Trans. Comput.*, vol. 42, no. 4, pp. 427–432, Apr. 1993.
- [Con04] R. Conway and J. Nelson, "Improved RNS FIR filter architectures," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 51, no. 1, pp. 26–28, Jan. 2004.
- [Con99] R. Conway and J. Nelson, "Fast converter for 3 moduli RNS using new property of CRT," *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 852-860, Aug. 1999.
- [Dim93] G. Dimauro, S. Impedovo, and G. Pirlo, "A new technique for fast number comparison in Residue Number System," *IEEE Trans. Comput.*, vol. 42, No. 5, pp. 608-612, May 1993.
- [Efs11] C. Efstathiou, K. Pekmestzi, N. Axelos, "On the design of modulo 2^n+1 multipliers,"

14th Euromicro Conference on Digital System Design (DSD), Oulu, Finland, Aug. 2011, pp. 453-459.

[Efs12] C. Efstathiou, N. Moschopoulos, K. Tsoumanis, K. Pekmestzi, "On the design of configurable modulo $2^n \pm 1$ residue generators", in *Proc. of 15th Euromicro Conference on Digital System Design (DSD)*, Izmir, Turkey, Sept. 2012, pp. 50-56.

[Etz80] M. Etzel and W.K. Jenkins, "Redundant residue number systems for error detection and correction in digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 5, pp. 538- 545, Oct 1980.

[Fau09] M. Faust and C. H. Chang, "Optimization of structural adders in fixed coefficient transposed direct form FIR filters," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, Taipei, Taiwan, 24-27 May 2009, pp. 2185–2188.

[Gar59] H. L. Garner, "The residue number system, " *IRE Trans. Electronic Computers*, vol. EC-8, pp. 140-147, June 1959.

[Gar99] A. Garcia and A. Lloris, "A look-up scheme for scaling in the RNS," *IEEE Trans. Comput.*, vol. 48, no. 7, pp. 748–751, Jul. 1999.

[Goh08] V. T. Goh and M. U. Siddiqi, "Multiple error detection and correction based on redundant residue number systems," *IEEE Trans. Commun.*, vol. 56, no. 3, pp. 325–330, Mar. 2008.

[Gol00] O. Goldreich, D. Ron, and M. Sudan, "Chinese remaindering with errors," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1330-1338, July 2000.

[Gri88] M. Griffin, F. Taylor, and M. Sousa, "New scaling algorithm for the Chinese remainder theorem," in *Proc. Conf. Rec. 22nd Asilomar Conf. Signals, Syst., Comput.*, 1988, pp. 375–378.

[Gri89] M. Griffin, M. Sousa, and F. Taylor, "Efficient scaling in the residue number system," in *Proc. 1989 Int. Conf. Acoust., Speech, Signal Process.*, Glasgow, U.K., May 1989, pp. 1075–1078.

[Han02] L. Hanzo, T. Liew, and B. Yeap, "Redundant Residue Number System Codes," in *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels*, 1st ed: Wiley-IEEE Press, 2002, pp. 257-316.

[Har11] N. Z. Haron and S. Hamdioui, "Redundant residue number system code for fault-tolerant hybrid memories," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 1, article 4, Jan. 2011.

[Hen99] H. Henkelmann and W. Anheier, "Implementation of sign detection in RNS using mixed radix representation," in *Proc. IEEE Int. Conf. Electron., Circuits and Syst.*, Paphos, Cyprus, vol. 1, Sep. 1999, pp. 323-326.

[How06] H. T. How, T. H. Liew, E.-L. Kuan, and L.-L. Yang, "A redundant residue number system coded burst-by-burst adaptive join-detection based CDMA speech transceiver, " *IEEE Trans. Veh. Technol.*, vol. 55, no. 1, pp. 387-396, Jan. 2006.

[Iwa09] H. Iwai, "Roadmap for 22 nm and beyond," *Microelectron. Eng.*, vol. 86, no. 7-9, pp. 1520-1528, Jul.-Sept. 2009.

[Jen83] W. K. Jenkins, "The design of error checkers for self-checking residue number arithmetic," *IEEE Trans. Comput.*, pp. 388-396, Apr. 1983.

[Jen88] W. K. Jenkins and E. J . Altman, "Self-checking properties of residue number error checkers based on mixed radix conversion," *IEEE Trans. Circuits Syst.*, pp. 159-167, Feb. 1988.

- [Jen93] W. K. Jenkins and A. J. Mansen, "Variable word length DSP using serial-by-modulus residue arithmetic," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Minneapolis, MN, USA, Apr. 1993, vol. 3, pp. 89–92.
- [Jul78] G. A. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Trans. Comput.*, vol. C-27, no. 4, pp. 325–336, Apr. 1978.
- [Kat96] R. S. Katti, "A new residue arithmetic error correction scheme," *IEEE Trans. Comput.*, vol. C-45, no. 1, pp. 13–19, Jan. 1996.
- [Kel00] T. Keller, T.-H. Liew, and L. Hanzo, "Adaptive redundant residue number system coded multicarrier modulation," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 11, pp. 2292–2301, Nov. 2000.
- [Kon09] Y. Kong and B. Phillips, "Fast scaling in the residue number system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 443–447, Mar. 2009.
- [Kri92] H. Krishna, K. Y. Lin, and J. D. Sun, "A coding theory approach to error control in redundant residue number systems. Part I: Theory and single error correction," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 8–17, Jan. 1992.
- [Kuc04] D. V. Ponomarev, G. Kucuk, O. Ergin, and K. Ghose, "Energy efficient comparators for superscalar datapaths," *IEEE Trans. Comput.*, vol. 53, no. 7, pp. 892–904, Jul. 2004.
- [Lie06] T. H. Liew, L.-L. Yang, and L. Hanzo, "Systematic redundant residue number system codes: analytical upper bound and iterative decoding performance over AWGN and Rayleigh channels," *IEEE Trans. Commun.*, vol. 54, no. 6, pp. 1006–1016, Jun. 2006.
- [Liu04] Y. Liu and E. M. K. Lai, "Design and implementation of an RNS-based 2-D DWT processor," *IEEE Trans. Consumer Electronics*, vol. 50, no. 1, pp. 376–385, Feb. 2004.

- [Low12] J. Y. S. Low and C. H. Chang, "A VLSI efficient programmable power-of-two scaler for $\{2^n-1, 2^n, 2^n+1\}$ RNS," *IEEE Trans. on Cir. and Syst.I*, vol. 59, no. 12, pp. 2911-2919, Dec. 2012.
- [Low13] J. Y. S. Low and C. H. Chang, "A new approach to the design of efficient residue generators for arbitrary moduli, " *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 9, pp. 2366-2374, Sept. 2013.
- [Lu92] M. Lu and J. Chiang, "A novel division algorithm for the Residue Number System," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 1026-1032, Aug. 1992.
- [Ma10] S. Ma, J. Hu, Y. Ye, L. Zhang, and X. Ling, "A 2^n scaling scheme for signed RNS integers and its VLSI implementation," *Sci. China, Inf. Sci.*, vol. 53, no. 1, pp. 203–212, Jan. 2010.
- [Maj11] P. Maj and G. S. Rath, "A novel design approach for low pass finite impulse response filter based on residue number system," in *Proc. 3rd Int. Conf. on Electronics Computer Technology*, Kanyakumari, India, pp. 74-78, Apr. 2011.
- [Man72] D. Mandelbaum, "Error correction in residue arithmetic," *IEEE Trans. Comput.*, vol. C-21, pp.538-545, June 1972.
- [Man76] D. Mandelbaum, "On a class of arithmetic codes and a decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, pp. 85-88, Jan 1976.
- [Mey03] U. Meyer-Baese and T. Stouraitis, "New power-of-2 RNS scaling scheme for cell-based IC design," *IEEE Trans. VLSI Syst.*, vol. 11, no. 2, pp. 280-283, Apr. 2003.

[Mil85] D. D. Miller, R. E. Altschul, J. R. King and J. N. Polky, “Analysis of the residue class core function of Akushskii, Burcev and Pak,” in *Residue Number System Arithmetic, Modern Applications in Digital Signal Processing*, M. A. Soderstrand, W. C. Jenkins, G. A. Jullien, and F. J. Taylor, eds. New York, IEEE Press, paper 7-2, 1985, pp. 390-401.

[Moh02] P. V. A. Mohan, *Residue Number Systems: Algorithms and Architectures*. Boston, Mass.: Kluwer Academic Publishers, 2002.

[Mol10] A.S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, S. Timarchi, “Efficient reverse converter designs for the new 4-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ based on New CRTs,” *IEEE Trans. on Circuits and Syst.-I*, vol. 57, no. 4, pp.823-835, Apr. 2010.

[Mur12] R. Muralidharan and C. H. Chang, “Area-power efficient modulo 2^n-1 and modulo 2^n+1 multipliers for $\{2^n-1, 2^n, 2^n+1\}$ based RNS,” *IEEE Trans. Circuits and Syst.-I* vol. 59, no. 10, pp. 2263 – 2274, Oct. 2012.

[Nel90] V. P. Nelson, “Fault-tolerant computing: Fundamental concepts,” *Computer*, vol. 23, no. 7, pp. 19–25, Jul. 1990.

[Ok194] V. G. Oklobdzija, “An algorithmic and novel design of a leading zero detector circuit: Comparison with logic synthesis,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 1, pp. 124–128, Mar. 1994.

[Omo07] A. R. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*. London, U.K.: Imperial College Press, 2007.

[Ort92] G. A. Orton, L. E. Peppard, and S. E. Tavares, “New fault tolerant techniques for residue number systems,” *IEEE Trans. Comput.*, vol. 41, no. 11, pp. 1453-1464, Nov. 1992.

- [Par09] B. Parhami, "Efficient hamming weight comparators for binary vectors based on accumulative and up/down parallel counters," *IEEE Trans. Circuits Syst.*, vol. 56, no. 2, pp. 167–171, Feb. 2009.
- [Par97] B. Parhami, "Modular reduction by multi-level table lookup," in *Proc. IEEE 40th Midwest Symp. Circuits Syst.*, Sacramento, CA, Aug. 1997, vol. 1, pp. 381–384.
- [Pet13] H. Pettenghi, R. Chaves, and L. Sousa, "RNS reverse converters for moduli sets with dynamic ranges up to $(8n+1)$ -bit," *IEEE Trans. Circuits and Syst.-I*, vol. 60, no. 6, pp. 1487–1500, Jun. 2013.
- [Pie10a] S. J. Piestrak, "Design of cost-efficient multipliers modulo 2^a-1 ," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, Paris, France, May 2010, pp. 4093 - 4096.
- [Pie10b] S. J. Piestrak, S. Pillement, and O. Sentieys, "Designing efficient codecs for bus-invert Berger code for fully asymmetric communication," *IEEE Trans. Circuits and Syst. II: Express Briefs*, vol. 57, no. 10, pp. 777–781, 2010.
- [Pon08] S. Pontarelli, G. C. Cardarilli, M. Re and A. Salsano, "A novel error detection and correction technique for RNS based FIR filters," *IEEE Int. Symp. Defect and Fault Tolerance of VLSI Systems*, Boston, MA, Oct. 2008, pp. 436-444.
- [Ram03] J. Ramirez, U. Meyer-Base, F. Taylor, A. Garcia and A. Lloris, "Design and implementation of high-performance RNS wavelet processors for custom IC technologies," *J. VLSI Signal Processing*, vol. 34, no. 3, pp. 227-237, July 2003.
- [Ram83] V. Ramachandran, "Single residue error correction in residue number systems," *IEEE Trans. Comput.*, pp. 504-507 May 1983.

- [Sha07] T. K. Shahana, R.K. James, B.R. Jose, K.P. Jacob, and S. Sasi, "Performance analysis of FIR digital filter design: RNS versus traditional," *Int. Symp. Communications and Information Technologies (ISCIT)*, Sydney, Australia, Oct. 2007, pp.1-5.
- [She04] M.-H. Sheu, S.-H. Lin, Y.-T. Chen, and Y.-C. Chang, "High-speed and reduced-area RNS forward converter based on $(2n-1, 2n, 2n+1)$ moduli set," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Tayi, Paraguay, Dec. 2004, pp. 821–824.
- [She08] Y. Sheng and W. Wang, "Design and implementation of compression algorithm comparator for digital image processing on component," in *Proc. 9th Int. Conf. Young Comput. Sci.*, Nov. 2008, pp. 1337–1341.
- [She89a] M. A. P. Shenoy and R. Kumaresan, "A fast and accurate RNS scaling technique for high speed signal processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 6, pp. 929–937, Jun. 1989.
- [She89b] M. A. P. Shenoy and R. Kumaresan, "Fast base extension using a redundant modulus in RNS," *IEEE Trans. Comput.*, vol. 38, no. 2, pp. 292-297, Feb. 1989.
- [Sou07] L. Sousa, "Efficient method for magnitude Comparison in RNS based on two pairs of conjugate moduli," *Proc. 18th IEEE Symp. Computer Arithmetic*, Montpellier, France, Jun. 2007, pp. 240-250.
- [Sto03] J. Stolfi and L.H. de Figueiredo, "An Introduction to Affine Arithmetic," *TEMA Tend. Mat. Apl. Comput.*, vol. 4, no. 3, pp. 297-312, 2003.
- [Su90] C. C. Su and H. Y. Lo, "An algorithm for scaling and single residue error correction in residue number systems," *IEEE Trans. on Computers*, vol. 39, no. 8, pp. 1053-1064, Aug. 1990.

- [Sun92] J. D. Sun and H. Krishna, "A coding theory approach to error control in redundant residue number systems. Part II: Multiple error detection and correction," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 18-34, Jan. 1992.
- [Suz07] H. Suzuki, C. H. Kim, and K. Roy, "Fast tag comparator using diode partitioned domino for 64-bit microprocessor," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 2, pp. 322–328, Feb. 2007.
- [Sza67] N. S. Szabó and R. I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York: McGraw-Hill, 1967, pp. 27-32.
- [Tan10] Y. Tang, E. Boutillon, C. Jégo and M. Jézéquel, "A new single-error correction scheme based on self-diagnosis residue number arithmetic," *Design and Architecture for Signal and Image Processing Conf.*, Edinburgh, Scotland, Oct. 2010, pp. 27-33.
- [Tay13] T. F. Tay, C. H. Chang, and J. Y. S. Low, "Efficient VLSI implementation of 2^n scaling of signed integer in RNS $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 10, pp. 1936–1940, Oct. 2013.
- [Tay14] T. F. Tay and C. H. Chang, "New algorithm for signed integer comparison in four-moduli superset $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$," accepted for publication in *2014 IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS)*.
- [Tay90] F. J. Taylor, "An RNS discrete Fourier transform implementation," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 38, no. 8, pp. 1386-1394, Aug. 1990.
- [Tin04] Goh, M. Tinauli and M. U. Siddiqi, "A novel error correction scheme based on the Chinese remainder theorem," Ninth Int. Conf. Communications Systems, Singapore, Sept. 2004, pp. 461-465.

[Toi06] T. Toivonen and J. Heikkilä, "Video filtering with Fermat number theoretic transforms using residue number system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 92–101, Jan. 2006.

[Tom08] T. Tomczak, "Fast sign detection for RNS (2^n-1 , 2^n , 2^n+1)," *IEEE Trans. Circuits and Syst. I*, vol. 55, no. 6, pp. 1502-1511, July 2008.

[Tso13] K. Tsoumanis, C. Efstathiou, N. Moschopoulos and K. Pekmestzi, "On the design of modulo $2^n \pm 1$ residue generators," *IEEE 21st Int. Conf. on Very Large Scale Integr. (VLSI-SoC)*, Istanbul, Turkey, Oct. 2013, pp.33-38.

[Ulm98] Z. D. Ulman and M. Czyzak, "Highly parallel, fast scaling of numbers in nonredundant residue arithmetic," *IEEE Trans. Signal Process.*, vol. 46, no. 2, pp. 487–496, Feb. 1998.

[Ver02] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2^n+1 adder design," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1389-1399, Dec. 2002.

[Vu85] T. V. Vu, "Efficient implementations of the Chinese Remainder Theorem for sign detection and residue decoding," *IEEE Trans. Comput.*, vol. C-34, no. 7, pp. 646-651, July 1985.

[Wan00] Y. Wang, "Residue-to-binary converters based on new Chinese remainder theorems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 3, pp. 192–206, Mar. 2000.

[Wan95] S. S. Wang and M. Y. Shau, "Single residue error correction based on K -term m_j -projection," *IEEE Trans. Comput.*, vol. 44, no. 1, pp. 129-131, Jan. 1995.

[Wan99] Y. Wang, X. Song, and M. Aboulhamid, "A new algorithm for RNS magnitude comparison based on new Chinese Remainder Theorem II," *Proc. 9th Great Lakes Symp. VLSI*, Michigan, USA, Mar. 1999, pp. 362-365.

[Wat66] R. W. Watson and C. W. Hastings, "Self-checked computation using residue arithmetic," *Proc. IEEE*, vol. 54, pp. 1920-1931, Dec. 1966.

[Yan02] L.-L. Yang and L. Hanzo, "A residue number system based parallel communication scheme using orthogonal signaling—part I: system outline," *IEEE Trans. Veh. Technol.*, vol. 51, no. 6, pp. 1534-1546, Nov. 2002.

[Yat13] V. Yatskiv, N. Yatskiv, J. Su, A. Sachenko and Z. Hu, "The use of modified correction code based on residue number system in WSN," *IEEE Int. Conf. Intelligent Data Acquisition and Advanced Computing Syst. (IDAACS) 2013*, Berlin, Germany, Sept. 2013, pp. 513-516.

[Yau73] S. S.-S. Yau and Y.-C. Liu, "Error correction in redundant residue number systems," *IEEE Trans. Comput.*, vol. C-22, pp. 5-11, Jan. 1973.

[Yau73] S. S.-S. Yau and Y.-C. Liu, "Error correction in redundant residue number systems," *IEEE Trans. Comput.*, vol. C-22, pp. 5-11, Jan. 1973.

[Ye08] Y. Ye, S. Ma, and J. Hu, "An efficient 2^n RNS scaler for moduli set $\{2^n-1, 2^n, 2^n+1\}$," in *IEEE Symp. Inf. Sci. Eng. (ISISE)*, Shanghai, China, Dec. 2008, pp. 511–515.

[Yu02] Z. Yu, M. Yu, K. Azadet, A. N. Willson, "A low power adaptive filter using dynamic reduced 2's-complement representation", in *Proc. IEEE Custom Integrated Circuits Conf.*, Orlando, FL, pp. 141-144, May 2002.

[Zar10] B. Zarei, M. Askarzadeh, N. Derakhshanfard, and M. Hosseinzadeh, "A High-Speed Residue Number Comparator For the 3-Moduli Set $\{2^n-1, 2^n+1, 2^n+3\}$," *Int. Symp. Signals*

Systems and Electronics, Nanjing, China, Sept. 2010, pp. 1-4.

[Zha12] S. Zhang, L.-L. Yang, and Y. Zhang, "Redundant residue number system assisted multicarrier direct-sequence code-division dynamic multiple access for cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 61, No. 3, pp. 1234 - 1250, Mar. 2012.

[Zim99] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, Adelaide, Australia, Apr. 1999, pp. 158-167.