www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

UNIVERSITY FOR DEVELOPMENT STUDIES

OVERFLOW DETECTION AND CORRECTION TECHNIQUES IN RNS ARITHMETIC COMPUTATIONS

PETER AWON-NATEMI AGBEDEMNAB

2015

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

UNIVERSITY FOR DEVELOPMENT STUDIES

OVERFLOW DETECTION AND CORRECTION TECHNIQUES IN RNS ARITHMETIC COMPUTATIONS

BY

PETER AWON-NATEMI AGBEDEMNAB
(BSc. Computer Science)
(UDS/MM/0020/13)

THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS, FACULTY OF MATHEMATICAL SCIENCES, UNIVERSITY FOR DEVELOPMENT STUDIES, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE DEGREE IN COMPUTATIONAL MATHEMATICS

JULY, 2015

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

DECLARATION

Student

I hereby declare that this thesis is the result of my own original work and that no part of it has been presented for another degree in this University or elsewhere.

Candidate's Signature:..... Date: 01/11/15

Name: **Peter Awon-natemi Agbedemnab**

Supervisor

I hereby declare that the preparation and presentation of the thesis was supervised in accordance with the guidelines on supervision of thesis laid down by the University for Development Studies.

Supervisor's Signature:..... Date: 05/11/15

Name: **Dr. Edem Kwedzo Bankas**

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

ABSTRACT

There is speed limitation on hardware built on Weighted Number Systems (WNS) due to carry propagation. In recent years, attempts have been made to circumvent the speed limitation imposed on WNS in arithmetic operations by investigating into number systems such as Residue Number System (RNS) which has special carry characteristics for parallel computations. RNS is carry-free in nature and is able to support parallel and high speed arithmetic such as addition and multiplication. Overflow detection is one of the difficult operations of RNS; but overflow can lead to the representation of incorrect values in the RNS system if not detected. Schemes that will be able to detect and correct overflow during arithmetic operations will boost research in the RNS. This thesis presents three schemes for detecting and correcting overflow in RNS arithmetic computations such addition: Two efficient algorithms for RNS overflow detection and correction for a generalised moduli set $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$, $\alpha = 1, 2, \ldots$ is presented. By taking $\alpha = 1$, the algorithms are applied to the popular moduli set $\{2^n-1, 2^n, 2^n+1\}$ for implementation and comparison. This is done by first devising a novel technique based on the Chinese Remainder Theorem (CRT) and another based on the Mixed Radix Conversion (MRC) method by performing partial RNS-binary conversion. The other technique also based on the MRC for the reverse conversion is applied on the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$ with a redundant modulus 2, which assists in detecting overflow during addition operations in the given moduli set. All the proposed schemes are able to prevent the representation of illegitimate numbers in the RNS system as if they were legitimate numbers thus correcting overflow. The schemes are demonstrated theoretically to be more efficient than similar state-of-the art schemes.

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

ACKNOWLEDGEMENT

I am very grateful to my supervisor, Dr. Edem Kwedzo Bankas, Head of Department (HoD), Computer Science Department, for is his mentorship and high degree of scrutiny which has brought me this far. His down-to-earth nature endeared me more to him and made our relationship cordial even outside of the work.

I will like to single out Dr. Mohammed Ibrahim Daabo, a lecturer in the Department of Computer Science, for his consistent interest in the status of my work. His constant reminders kept me on my toes, and I have always been ready to run if the need be. My sincere thanks to him. I also appreciate the inputs by my HoD, Ing. Dr. Ibrahim Yakubu Seini, in helping to shape up my work in order to make it an excellent piece of document.

My wife, Safia Haruna Aweh and my son Palti Awon-natemi have been a source of strength to me during this time of my life. May the Good Lord continues His unflinching favour on them so that their lives will continue to be a source of strength to me. My Mum and Siblings have also been very supportive. Thank you very much.

My friends; Mr. Akandawen Martin, Mr. Ayikpien John Yaw (Sandema Senior High/Technical School), Mr. Barik Abdul-Alhassan, a lecturer in the Computer Science Department, Mr. Bugli Clifford and all the staff of the Computer Science Department who have supported or encouraged me in one way or the other, I say thank you very much. My colleagues, Khalid, Timothy, Daniel and Osman have also been nice and I wish them all well in their endeavours.

www.udsspace.uds.edu.gh

Most importantly, is the GRACE, FAVOUR, MERCY and WISDOM that I have enjoyed throughout my life from the Almighty God; I owe tonnes of gratitude to Him and my whole life will continue to be of thanksgiving to Him.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

DEDICATION

To the memory of my late friend and brother, Hillary Siewobr.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

TABLE OF CONTENTS

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

LIST OF TABLES

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

LIST OF FIGURES

www.udsspace.uds.edu.gh

LIST OF ACRONYMS

CE	Cost Effective
CRT	Chinese Remainder Theorem
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DR	Dynamic Range
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GCD	Greatest Common Divisor
HS	High Speed
LSB	Least Significant Bit
MEP	Magnitude Evaluation Part
MRC	Mixed Radix Conversion
MRDs	Mixed Radix Digits
MSB	Most Significant Bit
OCP	Overflow Correction Part
ODP	Overflow Detection Part
ODU	Overflow Detection Unit
PCP	Partial Conversion Part
PRC	Partial Reverse Converter
RC	Reverse Converter
RNS	Residue Number System
VLSI	Very Large-Scale Integration
WNS	Weighted Number System

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

CHAPTER ONE

INTRODUCTION

1.0 Introduction

UNIVERSITY FOR DEVELOPMENT STUDIES

A number system is a language for representing numbers. It is defined by the set of values that each digit can assume and by the interpretation rule that defines the mapping between the sequences of digits and their numerical values (Kettani, 2006). The main task of computers is computing, which deals with numbers all the time. Numbers are the basis and object of computer operations (Lu, 2004). There are various forms of number systems: Weighted and Non-weighted. The Weighted Number System (WNS) is composed of a set of weights w_i such that, any number X can be expressed in the form: $X = \sum_{i=1}^{n} a_i w_i$ where a_i are the set of permissible digits. The values of w_i can be successive powers of the same number in which case the system is said to have fixed radix, however, if the weights are not the powers of the same radix then it is said to have mixed radix (Parhami, 2000).

Everyday number systems that are being used include the decimal (base 10) for easy counting and the Babylonians number system (base 60) which makes dealing with time easy (Parhami, 2000). Other systems that are used extensively in computer work are binary (base 2) in particular and octal (base 8) and hexadecimal (base 16); all these systems belong to the category of WNS. The WNS possesses features that makes it useful in computation:

(i) Relative-magnitude comparison of two numbers can be mechanised easily.

(ii) Multiplication or division in binary and decimal is achieved by merely shifting the digits in the storage registers.

www.udsspace.uds.edu.gh

(iii) Extending the range of the number system is easily realised by adding more digits position.

(iv) Overflow detection is also easily mechanised. Just to mention a few.

The attributes which lead to these advantages impose a limitation on the speed with which many arithmetic operations can be performed (Parhami, 2000 and Gbolagade, 2010).

In this system, true parallel arithmetic operations in which all the digits are processed simultaneously is not feasible because of carry propagation. Carry propagation results from the fact that, in this system, for all arithmetic operations, every digit of the result is a function of all digits of equal or lower significance. This characteristic makes it impracticable to implement parallel addition, subtraction, multiplication or division, and thus impose a limitation on the speed of arithmetic operations and ultimately hardware built on this system. In recent years, there have been two schools of thought as to how to circumvent this speed limitation namely, one, adding specialised look-ahead carry circuits and two, the use of number systems with special carry characteristics.

Residue Number System (RNS) belongs to the second school of thought. To enhance system performance, the RNS was proposed for computation-intensive application design because of its ability to support high-speed concurrent arithmetic (Sheu *et al.*, 2004). These RNS features have been put to good use in various digital signal processing applications (Nannarelli, Re, & Cardarilli, 2001). Today RNS is also regarded as one of the most popular techniques for reducing the power dissipation and the computation load in Very Large Scale Integrated Circuits (VLSI) system design (Stouratitis & Paliouras, 2001). RNS is a non-weighed number system with special carry characteristics and a potential that

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

results in high computations (Bankas & Gbolagade, 2013b). In RNS, addition, subtraction and multiplication are inherently carry-free (Bankas & Gbolagade, 2013c and Bankas & Gbolagade, 2014), for instance, each digit of the result is a function of only one digit from each operand, hence independent of all other digits. As a result of the carry-free property, it is feasible to mechanise operations such as addition, subtraction and multiplication.

UNIVERSITY FOR DEVELOPMENT STUDIES

1.1 Residue Number System

As stated earlier, RNS is a non-weighted number system that utilizes remainders to represent numbers. This number system is capable of supporting parallel, carry-free and high speed arithmetic (Bhardwaj, Srikanthan, & Clarke, 1999). This system also offers some useful properties such as parallelism, modularity, fault tolerance, and carry-free operations (Gbolagade *et al.*, 2010 and Bankas & Gbolagade, 2013a). It is very efficient in carrying out arithmetic operations like additions, subtractions and multiplications.

The speed of the arithmetic operations relies on the size of the numbers involved; smaller numbers imply faster operations. Since, the numbers used in this system are smaller, it is known for faster implementation of arithmetic operations, and hence it is very attractive (Gbolagade, 2010 and Chang, Low, & Yung, 2011). The system is applied in the fields of Digital Signal Processing (DSP) intensive computations like digital filtering, convolutions, correlations, Discrete Fourier Transform (DFT) computations, Fast Fourier Transform (FFT) computations and direct digital frequency synthesis (Mohan, 2007).

However, RNS has not found a widespread usage in general purpose computing due to the following difficult RNS arithmetic operations: overflow detection, magnitude comparison, sign detection, moduli selection, and data conversion (Omondi & Premkumar, 2007),

www.udsspace.uds.edu.gh

(Jaberipur & Ahmadifar, 2014 and Singh, 2008). Out of these numerous RNS challenges, Moduli selection, Data conversion, magnitude comparison, sign and overflow detection are the most critical issues (Bankas & Gbolagade, 2012 and Siewobr & Gbolagade, 2014a). Data conversion can be categorized into forward and reverse conversions. The forward conversion involves converting a binary or decimal number into its RNS equivalent whilst the reverse conversion involves converting the RNS number into binary or decimal (Bankas, 2013). Relatively, reverse conversion is more complex. A general structure of a typical RNS processor (Narayanaswamy, 2010), is shown in Figure 1.1.

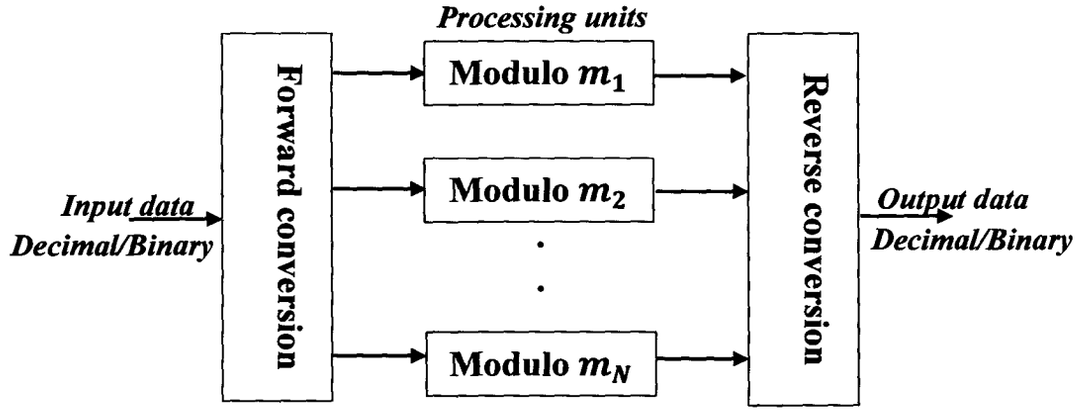

Figure 1. 1: General structure of an RNS-based processor

1.1.1 Overflow in RNS

Overflow, in general computing is said to occur if a calculated value is greater than its intended storage location in memory (Daabo, 2015). In RNS, overflow is a condition where a calculated number falls outside the valid dynamic range of a given RNS (Rouhifar *et al.*, 2011 and Siewobr & Gbolagade, 2011). Generally an overflow situation usually arises during addition and multiplication operations and failure to detect it will normally lead to improper or wrong representation of numbers and calculated results. Thus detecting overflow is one of the fundamental issues in the design of efficient RNS systems (Debnath

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

& Pucknell, 1978). For example in the popular moduli set $\{3, 4, 5\}$, the dynamic range is 60 and in representing numbers in this range, no number should be greater than 59. If we consider the decimal numbers 40 and 25 in this range, their sum is 65, which results in an overflow condition. In an RNS system, 40 and 25 are represented with respect to the moduli set $\{3, 4, 5\}$ as (1, 0, 0) and (1, 1, 0) respectively. Their sum with respect to $\{3, 4, 5\}$ is (2, 1, 0) which is equivalent to the decimal number 5. Surely, this is an error (overflow) because the sum of 40 and 25 cannot be the number 5. Such an error will be used in any system that the RNS is applied thereby leading to distorted or inefficient systems.

Effects of overflow in RNS systems

In DSP, an input signal is converted into a sequence of numbers that are the result of sampling and quantizing analog signals. In the process of extracting useful information, the signals are usually transformed from one type of representation into another type in which certain characteristics of the signal would become obvious (Omondi & Premkumar, 2007). An overflow in a representation therefore will lead to a wrong interpretation of the signal. Also, as computer chips become increasingly dense, the probability that some part may fail also increases; however, the large number of components also means that exhaustive testing is not likely to be practical. Therefore, computational integrity has become critical (Flynn & Huang, 2005 and Omondi & Premkumar, 2007). And so, there is the need to have fault tolerant systems. In communication, the multiple-access technique (easily achieved by use of RNS) used in communication engineering is one way of efficiently allocating a rare communication resource, namely, the radio spectrum. This technique becomes meaningful when a large number of users seek to communicate with each other simultaneously. This sharing of the frequency spectrum must be done in such a

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

way that it does not negatively affect the quality of performance of the system. Errors such as overflow during the sharing can affect the communication system. Finally, an image encryption system that contains errors will result in blurriness of the image. These are but a few of the RNS applications in which the occurrence of overflow can affect negatively.

1.1.2 Fundamentals of RNS

RNS is defined in terms of a set of relatively prime moduli set $\{m_i\}_{i=1,2,\ldots,n}$ such that the $\gcd(m_i, m_j) = 1$ for $i \neq j$, where gcd means the greatest common divisor of m_i and m_j, and $M = \prod_{i=1}^{n} m_i$, is the Dynamic Range (DR). The residues of a decimal number X can be obtained as $x_i = |X|_{m_i}$, thus X can be represented in RNS as $X = (x_1, x_2, \ldots, x_n)$, $0 \leq x_i \leq m_i$ which representation is unique for any integer $X \in [0, M-1]$. $|X|_{m_i}$ is the modulo operation of X with respect to m_i (Gbolagade *et al.*, 2010). Arithmetic operations such as (addition, subtraction and multiplication are performed totally in parallel on those independent residues as:

$X \circ Y = \{(x_1 \circ y_1), (x_2 \circ y_2), \ldots, (x_n \circ y_n)\}; \circ \equiv (+, -, \times)$ (Younes & Steffan, 2013).

The sign of the range of representable integers can be detected by dividing into two equal intervals as:

$$0 \leq X < \lfloor M/2 \rfloor \text{ (for positive integers) and}$$
$$\lfloor M/2 \rfloor \leq X < M \text{ (for negative integers)} \quad (1.1)$$

1.1.3 Advantages of RNS

Several advantages can be derived from the use of RNS. Some of them are outlined below:

(i) High speed arithmetic operation

When a weighted number is broken into residues, which are relatively smaller than their weighted equivalents, performing arithmetic operations tends to be faster.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

(ii) Parallel Arithmetic Operation

Arithmetic operations such as addition, subtraction, multiplication and division can be performed on residues simultaneously with each operation being independent of the other (Omondi & Premkumar, 2007 and Soderstrand *et al.*, 1986).

(iii) Carry Free Arithmetic

In RNS carries and borrows are not propagated and the result of an operation is with respect to a modulus as opposed to weighted number system where there are carries and borrows. To find the value of a negative number in a given modulus, the modulus is iteratively added to the given number until a positive number is obtained (Gbolagade, 2010).

(iv) Error Detection and Correction Capabilities

The modular nature of RNS makes error detection and correction easy as compared to other number systems. In operations such as addition, subtraction and multiplication, any particular digit of the resultants depends solely on the corresponding digits of the sub operation (Dugdale, 1992 and Gbolagade, 2010). Also multiplication is executed in a single step without any partial product and it is difficult for errors to propagate between parallel sub operations since they are independent of each other.

1.1.4 Challenges of RNS

Notwithstanding the numerous advantages associated to the use of RNS, it is confronted with a lot of challenges which make its application and implementation very difficult. Some of these challenges are outlined below:

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

(i) Data Conversion

The most direct way to convert from a conventional representation to a residue representation, a process known as forward conversion, is to divide by each of the given moduli and then collect the remainders. This however, is likely to be a costly operation if the number is represented in an arbitrary radix and the moduli are arbitrary. If on the other hand, the number is represented in radix-2 (or a radix that is a power of two) and the moduli are of a suitable form (for example $2^n - 1$), then the procedures can be implemented with more efficiency. The conversion from residue notation to a conventional notation, a process known as reverse conversion, is more difficult (conceptually, if not necessarily in the implementation) and so far has been one of the major impediments to the adoption and use of RNS (Omondi & Premkumar, 2007).

(ii) Overflow Detection

As mentioned earlier, Overflow is a condition where a number which falls outside the legitimate range of a particular RNS i.e.,$[0, M-1]$, $(M = \prod_{i=1}^{n} m_i)$ is represented as a legitimate RNS number. The general method of detecting overflow in RNS is by comparing the result of addition with one of the addends. If $X \geq 0$ and $Y < M$ then $(X + Y)$ mod M causes overflow if and only if the result is less than X (Theodore, 1989; Omondi & Premkumar, 2007 and Younes, 2013). One of the most efficient ways to detect overflow in RNS is via parity checking (Rouhifar *et al.*, 2011). It indicates whether an integer is even or odd. Suppose two integers (X, Y) have the same parity: $Z = X + Y$. An overflow occurs if Z is odd. Contrary, if (X, Y) have different parity, then an overflow occurs if Z is even. This technique

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

is one of the best and fastest suggested methods to detect the overflow in RNS. However, this technique is only suitable for moduli sets with odd dynamic range. But RNS systems that have even dynamic ranges, have more attractive features than those with odd dynamic ranges. This is because using (2^n) modulo which turns dynamic ranges to even, greatly simplifies and reduces the delay and complexity of the scheme (Gbolagade, 2010).

(iii) Sign Detection

In contrast to the conventional weighted number systems, the sign of a residue number is a function of all the residue digits of that number. Due to this, numbers represented in RNS are without the use of negative or positive signs thus making it virtually impossible to differentiate positive and negative numbers (Omondi & Premkumar, 2007).

(iv) Moduli Selection

The kind of moduli set that is used, determines the range of values that can be represented in RNS, the complexity of the arithmetic operation and also the speed of the arithmetic operations (Molahosseini *et al.*, 2010). The efficiency of RNS largely depends on the moduli set selected.

(v) Magnitude Comparison

It is very difficult to determine the magnitude of a residue number at a glance due to the fact that it is not weighted. This to some extent limits the operations that can be performed in RNS.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

(vi) Difficulty in carrying out complex arithmetic operations

It is practically easy when it comes to performing arithmetic operations such as addition and subtraction in RNS, this is not so in the case of division (Keir, Cheney, & Tannenbaum, 1962). The complexity associated with the division of numbers in RNS makes it difficult to implement.

1.1.5 Application of RNS

One of the advantages of RNS is its ability to carry out high speed computation as well as perform parallel arithmetic; this has led to its adoption in Digital Signal Processing (DSP) applications. RNS is used in DSP applications such as:

(i) Digital Filtering

A digital filter refers to an electronic device designed to operate on a sample discrete-time signal to enhance or reduce certain aspects of that signal. Digital filter microprocessors are constructed using RNS; it carries out intensive numerical operations (Etzel & Jenkins, 1980 and Younes, 2013). Digital filters make it possible to have many designs which are impracticable with analogue filters and find a whole range of applications in areas of cell phone and stereo receiver production (Omondi & Premkumar, 2007).

(ii) Discrete Cosine Transform (DCT)

This makes use of a sequence of finitely many data points expressed in terms of the sum cosine functions oscillating at different frequencies. DCT's are important to a number of applications in science and engineering from lossy compression of audio and images to spectral methods for the numerical solution of partial differential equations (Omondi & Premkumar, 2007).

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

(iii) Discrete Fourier Transform (DFT)

Here, a special kind of transform called the Discrete Fourier Transform is used in fourier analysis where one function is transformed into another called the domain representation or DFT for short of signal function. This is mostly a function in the time domain (Omondi & Premkumar, 2007 and Younes, 2013).

(iv) Fast Fourier Transform (FFT)

RNS is also used in Fast Fourier Transform algorithm that is used to efficiently compute the DFT and its inverses. A DFT breaks down or decomposes a sequence of values into components of different frequencies. This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. A FFT computes the same results more quickly (Omondi & Premkumar, 2007).

(v) Digital Communication

The application of RNS in digital communication is mainly for performing tasks such as direct digital synthesis using RNS processors.

(vi) Error Detection and Correction

The modular nature of RNS makes it a fault tolerant number system. This feature makes RNS a desirable tool for error detection and correction.

UNIVERSITY FOR DEVELOPMENT STUDIES

1.2 Problem Statement

Many researchers have proposed schemes for both odd and even dynamic range using techniques that do not require the full-reverse conversion process. Recently proposed RNS overflow detection algorithms still rely on the later (Younes & Steffan, 2013), and other costly and time consuming procedures such as base extension, group number and sign detection (Rouhifar *et al.*, 2011) and (Siewobr & Gbolagade, 2014b). From literature

www.udsspace.uds.edu.gh

available, most works on overflow detection are silent on how to correct overflow if it occurs. Failure to detect overflow in the RNS processor during operations such as addition and multiplication will lead to wrong representation and calculated results; detecting overflow alone is not enough. The ability to correct it in a more efficient way will lead to an enhancement of the RNS processor. The RNS will gain prominence in the world of computing if errors such as overflow can be detected and corrected as well.

This thesis presents efficient algorithms for RNS overflow detection and correction in RNS arithmetic computations such as addition.

UNIVERSITY FOR DEVELOPMENT STUDIES

1.3 Research Questions

As a result of the problems stated above, the following research questions are to be investigated;

(i) Can the CRT and MRC be simplified for a generalised moduli set $\{2^{an}-1, 2^{an}, 2^{an}+1\}$ with even dynamic range?

(ii) Can the simplification in (i) be simplified to the popular moduli set $\{2^{n}-1, 2^{n}, 2^{n}+1\}$ in order to design an overflow detection and correction scheme?

(iii) Can the MRC be simplified for the moduli set $\{2^{2n+1}-1, 2^{n}+1, 2^{n}-1\}$ in a manner that overflow detection and correction will not require full reverse conversion?

(iv) What will be the performance of the proposed algorithms in terms of hardware resources and speed compared to similar state-of-the art schemes?

www.udsspace.uds.edu.gh

1.4 Main Objective of the study

The main goal of the thesis is to propose efficient schemes to detect and correct overflow in RNS arithmetic computations.

1.4.1 Specific Objectives of the Study

The specific objectives of the thesis include but not limited to the following:

(i) Simplify the CRT and MRC for a generalised moduli set $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$ with even dynamic range.

(ii) Apply the simplification in (ii) to the popular moduli set $\{2^n-1,\ 2^n,\ 2^n+1\}$ in order to design overflow detection and correction schemes.

(iii) Simplify the MRC for the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$ in a manner that overflow detection and correction will not require full reverse conversion process.

(iv) Evaluate the performance of the proposed algorithms against the state of the art designs in terms of hardware resources and speed.

1.5 Significance of the Study

For a successful application of RNS, data conversion must be very fast so that the conversion overhead does not nullify the RNS advantages but the converted data may contain errors if overflow occurs and is not detected, more so, if it cannot be corrected. The effective design and implementation of schemes that can detect and correct overflow in RNS arithmetic computations will eliminate the errors associated with the conversion process and also build fault tolerant architectures. Proposing efficient reverse converters and overflow detection algorithms will therefore bring enhancement in speed and area consumption to DSP intensive computations like digital filtering, convolutions,

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

correlations, DFT, FFT computations, direct digital frequency synthesis, image processing and cryptography.

1.6 Organisation of the Study

The rest of the thesis is organised as follows: In chapter two, a background information on RNS as well as a review of related works on overflow detection will be presented. In chapter three, two techniques for detecting and correcting overflow during addition operations in the moduli set $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$ will be presented. For purpose of implementation, the popular moduli set $\{2^n-1, 2^n, 2^n+1\}$ thus taking $\alpha = 1$, will be used. The first technique is going to be based on the CRT and the second based on the MRC by using MRDs to evaluate the sign of the addends. In chapter four, a different method of detecting and correcting overflow in the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$ will also be presented. In this method, a redundant modulus 2 will be added to the given moduli set for detecting overflow during RNS addition. Finally, chapter five presents summary of findings, conclusions, recommendation and future work of the study.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

CHAPTER TWO

BACKGROUND INFORMATION AND REVIEW OF RELATED WORKS

2.0 Introduction

Notwithstanding the many advantages and vast application areas of the RNS, overflow is still one of the challenges of this system. This chapter presents some history and background information on RNS and overflow detection in RNS and reviews other selected research works on overflow detection.

2.1 History of RNS

A riddle posted in a book authored by a Chinese scholar called Sun Tzu in the first century was the first documented manifestation of RNS representation (Omondi & Premkumar, 2007 and Singh, 2008). The riddle is described by the following statements:

We have things of which we do not know the number:

If we count them by threes, the remainder is 2.

If we count them by fives, the remainder is 3.

If we count them by sevens, the remainder is 2.

How many things are there?

The answer is 23.

In other words, "What number yields the remainders 2, 3, and 2 when divided by 3, 5, and 7, respectively?" In modern terminology, 2, 3, and 2 are residues, and 3, 5, and 7, are moduli. Sun Tzu gave a rule, the Tai-Yen (Great Generalization) for the solution of his puzzle. In 1247 AD, another Chinese mathematician, Qin Jiushao, generalized the Great

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

Generalization into what is now known as the Chinese Remainder Theorem (CRT), a mathematical jewel (Omondi & Premkumar, 2007). The CRT provides an algorithmic solution of decoding the residue encoded number back into its conventional representation. This theorem is considered the cornerstone in realizing RNSs (Narayanaswamy, 2010). Encoding a large number into a group of small numbers results in significant speed up of the overall data processing. This fact encourages the implementation of RNS in some applications where intensive processing is inevitable.

In the 1950s, RNS was rediscovered by computer scientists, who sought to put it to use in the implementation of fast arithmetic and fault-tolerant computing (Omondi & Premkumar, 2007). Three properties of RNS make it well suited for these. The first is absence of carry-propagation in addition and multiplication, carry-propagation being the most significant speed-limiting factor in these operations. The second is that because the residue representations carry no weight-information, an error in any digit-position in a given representation does not affect other digit-positions. The third is that there is no significance-ordering of digits in an RNS representation, which means that faulty digit-positions may be discarded with no effect other than a reduction in dynamic range.

The new interest in RNS was not long-lived, for three main reasons: One, a complete arithmetic unit should be capable of at least addition, multiplication, division, square-root, and comparisons, but implementing the last three in RNS is not easy; two, computer technology became more reliable; and, three, converting from RNS notation to conventional notation, for "human consumption", is difficult. Nevertheless, in recent years there has been renewed interest in RNS. There are several reasons for this new interest,

www.udsspace.uds.edu.gh

including the following. A great deal of computing now takes place in embedded processors, such as those found in mobile devices, and for these high speed and low-power consumption are critical; the absence of carry-propagation facilitates the realization of high-speed, low-power arithmetic. Also, computer chips are now getting to be so dense that full testing will no longer be possible; so fault-tolerance and the general area of computational integrity have again become more important. Lastly, there has been progress in the implementation of the difficult arithmetic operations (Omondi & Premkumar, 2007). True, that progress has not been of an order that would justify a deluge of letters home; but progress is progress, and the proper attitude should be gratitude for whatever we can get. In any case, RNS is extremely good for many applications – such as digital signal processing, communications engineering, computer security (cryptography), image processing, speech processing, and transforms – in which the critical arithmetic operations are addition and multiplication (Younes, 2013).

UNIVERSITY FOR DEVELOPMENT STUDIES

2.2 The Concept of Overflow Detection

Overflow, in general computing is said to occur if a calculated value is greater than its intended storage location in memory. In RNS, overflow is said to be a condition where a calculated number falls outside the valid dynamic range of a given RNS (Siewobr & Gbolagade, 2011) and (Daabo & Gbolagade, 2012). Overflow detection is one of the fundamental issues in efficient design of RNS systems. In a more generalized approach, overflow occurs in the addition of two numbers X and Y, whenever $(X + Y)$ mod M is less than X, where $M = \prod_{i=1}^{n} m_i$ is the system dynamic range (Daabo & Gbolagade, 2014). Thus, the problem of overflow detection in RNS arithmetic is equivalent to magnitude comparison (Askarzadeh, Hosseinzadeh, & Navi, 2009). Another algorithm that has been

www.udsspace.uds.edu.gh

proposed for overflow detection in odd dynamic range M is a ROM-based algorithm and is called the parity checking technique. Parity indicates whether an integer number is even or odd. Given the operands X and Y which have the same parity, and $Z = X + Y$. The addition process is with overflow, if Z is an odd number (Shang, *et al.*, 2008). For signed overflow detection in RNS, overflow occurs when the sign of the sum of operands is different from the sign of the operands (Rouhifar *et al.*, 2011).

2.3 Traditional Methods of Overflow Detection

In the traditional approach, overflow detection is done by full reverse conversion (Siewobr & Gbolagade, 2014b). Thus, given the residue representation of a number, overflow can only be detected by fully converting the number back to its binary or decimal equivalent. Given the residues $x_i = |X|_{m_i}$, $i = 1,2,3, \ldots$, many techniques/algorithms have been devised to convert back to its binary equivalent. Popular among these residue-to-binary conversion techniques are the Chinese Remainder Theorem (CRT), Mixed Radix Conversion (MRC), and the recently modified CRT to CRT-I, CRT-II and CRT-III among others (Hosseinzadeh, Molahosseini, & Navi, 2009).

2.3.1 Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) can be used to backward convert the residue digits $(x_1, x_2, \ldots, x_n)$ of the moduli set $\{m_1, m_2, \ldots, m_n\}$ to its decimal number X to detect overflow as follows:

$$X = \left| \sum_{i=1}^{n} \ell_i \, |k_i x_i|_{m_i} \right|_M \tag{2.1}$$

where,

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$M = \prod_{i=1}^{N} m_i \; ; \quad \ell_i = \frac{M}{m_i} \; ; \; |k_i \times \ell_i|_{m_i} = 1$$

The main problem with the CRT is the involvement of large modulo M operation which is generally expensive.

2.3.2 Mixed Radix Conversion

The Mixed Radix Conversion (MRC) approach serves as an alternative method to the CRT as it does not involve the use of the large modulo$-M$ computation. This method also detect overflow by converting the residue digits $(x_1, x_2, \dots, x_n)$ of the moduli set $\{m_1, m_2, \dots, m_n\}$ to its decimal equivalent X as follows:

$$X = e_1 + e_2 m_1 + e_3 m_1 m_2 + \cdots + e_n m_1 m_2 m_3 \dots m_{n-1} \tag{2.2}$$

where $e_i, i = 1,2, \dots, n$ are the Mixed Radix Digits (MRDs) and computed as follows:

$$e_1 = x_1$$

$$e_2 = \left|(x_2 - e_1)|m_1^{-1}|_{m_2}\right|_{m_2}$$

$$e_3 = \left|\left((x_3 - e_1)|m_1^{-1}|_{m_3} - e_2\right)|m_2^{-1}|_{m_3}\right|_{m_3}$$

$$\vdots$$

$$e_n = \left|\left(\dots\left((x_3 - e_1)|m_1^{-1}|_{m_n} - e_2\right)|m_2^{-1}|_{m_n} - \cdots - e_{n-1}\right)|m_{n-1}^{-1}|_{m_n}\right|_{m_n} \tag{2.3}$$

The MRDs e_i are within the range $0 \leq e_i \leq m_i$, and a positive number, X, in the interval $[0, M]$ can be uniquely represented. It can be seen that the MRC method has an advantage over the CRT approach since it only uses mod-m_i instead of mod-M as in the case of the CRT. However, MRC by its very nature involves sequential operations and can limit speed (Gbolagade *et al.*, 2010).

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

2.4 Dynamic Range Overflow

Most research work on overflow detection are based on processors built separately to detect multiplicative and additive overflow (Theodore, 1989). Invariably, most of the schemes adopted are still completely dependent on the CRT or the MRC and thus resulting in processors that are hardware intensive and slow in nature. Siewobr & Gbolagade, (2011) proposed an overflow detection algorithm for the moduli set $\{M, M+1\}$. In their proposal, the residue number (x_i, x_{i+1}) is obtained from $\{M, M+1\}$ and the validity of no overflow occurs when $x_i = x_{i+1}$. However building adders with the $M+1$ modulus is very expensive. The authors again developed an additive overflow detection algorithm that reduces the large modulo M to M_i by scaling M and integers X and Y with $m_1 = 2^n$. This approach is hardware intensive with increase in delay since the process involves scaling.

2.5 Other Approaches

Recently other techniques have been developed to detect overflow without necessarily completing the reverse conversion process; Askarzadeh *et al.*, (2009) proposed an algorithm to detect overflow in the moduli set $(2^n - 3, 2^n - 1, 2^n, 2^n + 1, 2^n + 3)$ by adding a redundant modulus 2 to this moduli set and making use of ROM and XOR gates. It was demonstrated that their proposed algorithm could be implemented by a limited number of components and had less delay compared to previous works. Rouhifar *et al.*, (2011) presented a novel method for detecting overflow in the moduli set $(2^n - 1, 2^n, 2^n + 1)$ based on group of numbers. In their proposed method, numbers $[0, M-1]$ are distributed among several groups. Then, by using the groupings, they were able to diagnose in the process of addition of two numbers, whether overflow has occurred or not without

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

doing a complete comparison or need to use the residue to binary converter. Younes & Steffan, (2013) evaluated the sign of the sum of two numbers X and Y and used it to detect overflow. They also suggested that whenever overflow occurs, adding back M (i.e. dynamic range) to the result will give the correct value and thus called it overflow correction technique. The authors adopted a residue-to-binary converter proposed by (Piestrak, 1995). Siewobr & Gbolagade, (2014b) presented a generalised scheme for detecting overflow in RNS, followed by a simplified Operands Examination Method for overflow detection for the moduli set $(2^n - 1, 2^n, 2^n + 1)$. Their proposed method detected overflow in RNS addition of two numbers without pre-computing their sum. All these schemes either relied on complete reverse conversion process as in the case of (Younes & Steffan, 2013), or other costly and time consuming procedures such as base extension, group number and sign detection as in (Rouhifar *et al.*, 2011 and Siewobr & Gbolagade, 2014b).

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

CHAPTER THREE

OVERFLOW DETECTION AND CORRECTION SCHEMES FOR MODULI SETS OF THE FORM $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$ DURING RNS ADDITION

3.0 Introduction

In this chapter, two efficient algorithms for RNS overflow detection and correction for generalised moduli sets of the form $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$, $\alpha = 1, 2, \ldots$ are proposed. But in order to implement and compare the proposed algorithms, $\alpha = 1$ is considered and applied to the popular moduli set $\{2^n-1, 2^n, 2^n+1\}$ for implementation and comparison. Thus, the choice of α values will determine the range of legitimate RNS numbers that can be represented.

3.1 RNS Additive Overflow Detection and Correction Based on the Chinese Remainder Theorem

In this section, a novel method for detecting overflow as well as preventing the representation of illegitimate numbers as if they are legitimate numbers in the DR (thus correcting overflow) is presented. The ensuing method will be based on a simplification the Chinese Remainder Theorem to achieve a partial reverse conversion of two RNS numbers X and Y and then a novel algorithm developed to check the occurrence or otherwise of overflow during the addition of these numbers; and if overflow occurs, the technique avoids the representation of wrong values and gives the correct values.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

3.1.1 Proposed Method

Given the RNS numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$ with respect to the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, where $m_1 = 2^n - 1$, $m_2 = 2^n$ and $m_3 = 2^n + 1$ we have from equation (2.1):

$$\ell_1 = 2^n(2^n + 1); \quad \ell_2 = (2^{2n} - 1); \quad \ell_3 = 2^n(2^n - 1) \tag{3.1}$$

Lemma 3.1: For the given moduli set, we have, (Bankas & Gbolagade, 2013b):

$$|k_1|_{m_1} = |2^{n-1}|_{m_1} \tag{3.2}$$

$$|k_2|_{m_2} = |-1|_{m_2} \tag{3.3}$$

$$|k_3|_{m_3} = |-2^{n-1}|_{m_3} \tag{3.4}$$

Proof: If it can be demonstrated that $|\ell_i \times k_i|_{m_i} = 1$ then $|k_i|_{m_i}$ is the multiplicative inverse of ℓ_i mod m_i, $i = 1,2,3$:

Thus for (3.2), $|2^n(2^n + 1) \times (2^{n-1})|_{2^n-1} = |(2^n + 1) \times (2^{n-1})|_{2^n-1}$

$$= |2 \times (2^{n-1})|_{2^n-1}$$

$$= |2^n|_{2^n-1}$$

$$= |1|_{2^n-1} = 1$$

Also for (3.3), $|(2^{2n} - 1) \times (-1)|_{2^n} = |(-1) \times (-1)|_{2^n}$

$$= |1|_{2^n} = 1$$

Lastly for (3.4), $|2^n(2^n - 1) \times (-2^{n-1})|_{2^n+1} = |(2^n - 1) \times (2^{n-1})|_{2^n+1}$

$$= |-2 \times (2^{n-1})|_{2^n+1}$$

$$= |(-1) \times (-1)|_{2^n+1}$$

$$= |1|_{2^n+1} = 1 \qquad \blacksquare$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Theorem 3.1: For the given moduli set, any RNS number X can be represented as;

$$X = 2^n\rho + x_2 \tag{3.5}$$

where,

$$\rho = \left|\left|(2^n x_1 + x_1)2^{n-1}\right|_{2^{2n}-1} + \left|-2^n x_2\right|_{2^{2n}-1} + \left|-x_3\right|_{2^{2n}-1} + \left|+2^{n-1}x_3\right|_{2^{2n}-1}\right|_{2^{2n}-1} \tag{3.6}$$

Proof: Substituting equations (3.1) through to (3.4) into equation (2.1) and factorizing out 2^n we obtain (3.5).

From (3.5), let X and Y be two RNS numbers such that their sum is Z. Which implies that:

$$X = 2^n\rho_x + x_2 \tag{3.7}$$

$$Y = 2^n\rho_y + y_2 \tag{3.8}$$

$$Z = 2^n(\rho_x + \rho_y) + x_2 + y_2$$

$$= 2^n K + \mu \tag{3.9}$$

Where $K = \rho_x + \rho_y$ and $\mu = x_2 + y_2$

Let

$$\beta = \begin{cases} 0; & \mu < 2^n \\ 1; & \mu \geq 2^n \end{cases} \tag{3.10}$$

Theorem 3.2: Given any two RNS numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$, overflow occurs if and only if

$$K \geq 2^{2n} - 1 \tag{3.11}$$

or

$$K = 2^{2n} - 2 \text{ and } \beta = 1 \tag{3.12}$$

Proof: Assume (3.11) holds true; then for (3.9)

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$Z \geq 2^n(2^{2n} - 1) + \mu$$

$$\geq M + \mu \tag{3.13}$$

Which is outside the legitimate range, i.e. $[0, M - 1]$, hence overflow will occur.

Furthermore, if (3.12) holds true then (3.9) can be rewritten as

$$Z = 2^n(2^{2n} - 2 + 1)$$

$$= M, \tag{3.14}$$

which is also outside the legitimate range, therefore overflow will occur. Hence proofed.

From equation (3.9), Z will be the correct result of summing X and Y whether overflow occurs or not in the given moduli set, but will be out of the range in $[0, M - 1]$ if either (3.11) or (3.12) holds; therefore K should be added to the DR to be $[0, M + K - 1]$ in order to legitimize Z.

3.1.2 Hardware Implementation

Equation (3.6) can further be simplified as follows

$$\rho = |\varphi_1 + \varphi_2 + \varphi_3 + \varphi_4|_{2^{2n}-1} \tag{3.15}$$

where

$$\varphi_1 = |(2^n x_1 + x_1)2^{n-1}|_{2^{2n}-1} \tag{3.16}$$

$$\varphi_2 = |-2^n x_2|_{2^{2n}-1} \tag{3.17}$$

$$\varphi_3 = |-x_3|_{2^{2n}-1} \tag{3.18}$$

$$\varphi_4 = |2^{n-1} x_3|_{2^{2n}-1} \tag{3.19}$$

Now, we consider (3.15)-(3.19) and simplify them for implementation in a VLSI system. It is necessary to note that $x_{i,j}$ means the j-th bit of x_i.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Evaluation of φ_1

The residue x_1 can be represented as follows;

$$x_1 = x_{1,n-1} \dots x_{1,1}x_{1,0} \tag{3.20}$$

Thus,

$$|(2^n x_1 + x_1)2^{n-1}|_{2^{2n}-1} = \left|2^{n-1}\left(\underbrace{x_{1,n-1} \dots x_{1,0}\overbrace{0\dots0}^{n\ Bits}}_{2n-bits} + \underbrace{\overbrace{0\dots0}^{n\ Bits}x_{1,n-1} \dots x_{1,0}}_{2n}\right)\right|_{2^{2n}-1}$$

$$= \left|2^{n-1}\left(\underbrace{x_{1,n-1} \dots x_{1,1}x_{1,0}x_{1,n-1} \dots x_{1,1}x_{1,0}}_{2n-bits}\right)\right|_{2^{2n}-1}$$

$$= \overbrace{x_{1,0}x_{1,n-1} \dots x_{1,1}x_{1,0}}^{n+1\ Bits}\underbrace{x_{1,n-1} \dots x_{1,n+2}x_{1,1}}_{n-1} \tag{3.21}$$

Evaluation of φ_2*:*

The residue x_2 can be represented as follows;

$$x_2 = x_{2,n-1} \dots x_{2,1}x_{2,0} \tag{3.22}$$

Therefore,

$$|-2^n x_2|_{2^{2n}-1} = \overline{x}_{2,n-1} \dots \overline{x}_{2,1}\overline{x}_{2,0}\overbrace{11\dots11}^{n\ Bits} \tag{3.23}$$

Evaluation of φ_3 *and* φ_4:

The residue x_3 can be represented as follows;

$$x_3 = x_{3,n} \dots x_{3,1}x_{3,0} \tag{3.24}$$

Therefore,

$$\varphi_3 = |-x_3|_{2^{2n}-1} = \overbrace{11\dots11}^{n\ Bits}\underbrace{\overline{x}_{3,n-1} \dots \overline{x}_{3,1}\overline{x}_{3,0}}_{n\ Bits} \tag{3.25}$$

Again,

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$\varphi_4 = |2^{n-1}x_3|_{2^{2n}-1} = \underbrace{0x_{3,n} \dots x_{3,1}x_{3,0}}_{n+1\ Bits}\overbrace{00\dots00}^{n-1\ Bits} \tag{3.26}$$

Correction unit

In order to evaluate the sum Z, we further simplify equation (3.9).

$$Z = \tau + \mu \tag{3.27}$$

$$\tau = 2^n K = \underbrace{K_{2n}K_{2n-1} \dots K_1K_0\overbrace{00 \dots 0}^{n\ bits}}_{3n+1\ bits} \tag{3.28}$$

$$\mu = \underbrace{\mu_n\mu_{n-1} \dots \mu_1\mu_0}_{n+1\ bis} \tag{3.29}$$

Therefore,

$$Z = \underbrace{\tau_{3n}\tau_{3n-1} \dots \tau_1\tau_0 + \overbrace{00 \dots 0}^{2n\ bits}\mu_n\mu_{n-1} \dots \mu_1\mu_0}_{3n+1\ bits} \tag{3.30}$$

Implementation of equations (3.27) – (3.30) gives the correct result of Z whether overflow occurs or not.

3.1.3 Proposed Architecture

The floor function ρ, in (3.6) is computed according to equation (3.15) where all the parameters are defined in equations (3.16) – (3.19). For two numbers X and Y, ρ_x and ρ_y are the ρ values corresponding X and Y respectively and are computed separately. As shown in Figure 3.1, ρ is computed using CSAs 1and 2 and two regular $2n$-bit CPAs 1 and 2. The results of these CPAs are passed on to a multiplexer (MUX 1), which would then pass either of them down. MUX 1 will pass on the result of CPA 1 if the carry out of CSA 1 is a '0', otherwise the result of CPA 2 is passed on. ρ_x corresponding to the binary number X and ρ_y corresponding to the binary number Y are added using a regular $(2n + 1)\ bits$ CPA 3 in order to get K; at the same time, x_2 and y_2 is computed using a regular

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$(n+1)$ *bits* CPA 4 to obtain μ. A multiplexer (Mux 2) is used to select the value of β to be zero if the most significant bit (MSB) of μ is 0, otherwise, it selects one (1) if the MSB of μ is 1. This is shown in Figure 3.2 which is the overflow detection unit. CSAs 1 and 2 require an area of $2n\Delta_{FA}$ each as well as CPAs 1 and 2. In order to obtain ρ will require a total area of $8n\Delta_{FA}$. So for two numbers X and Y, the total area requirement will be $16n\Delta_{FA}$. CPA 3 demands an area of $(2n+1)\Delta_{FA}$ and CPA 4 also requires $(n+1)\Delta_{FA}$ of resources. Thus, the area requirement for the overflow detection component is $(3n+2)\Delta_{FA}$. Therefore, the total area requirement of the overflow detection scheme is $(19n+2)\Delta_{FA}$. Regarding the delay, each CSA (i.e. CSAs 1 and 2) impose a delay of D_{FA} while the CPA pair 1 and 2 impose a delay of $2nD_{FA}$ since they are in parallel, for two numbers this will become $4nD_{FA}$, thus delay imposed on computing ρ is $(4n+2)D_{FA}$. Also the two CPA pair 3 and 4 imposes a delay of $(2n+1)D_{FA}$ for the overflow detection unit. The delay required for the proposed scheme is $(6n+3)D_{FA}$. The correction unit (shown in Figure 3.3) uses a regular $(3n+1)$ *bits* CPA 5. The area requirement is $(3n+1)\Delta_{FA}$ and its delay is also $(3n+1)D_{FA}$. The schematic diagrams for the proposed scheme are shown in figures 3.1, 3.2 and 3.3.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

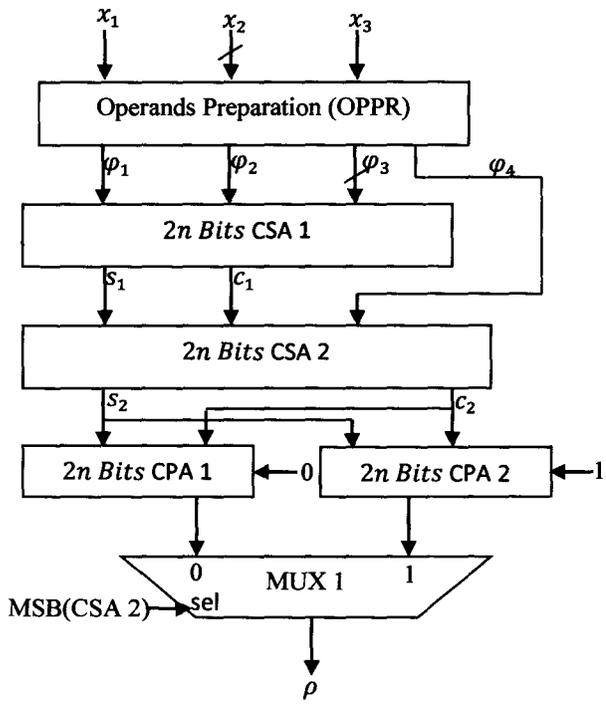

Figure 3.1: Block diagram of the partial reverse converter

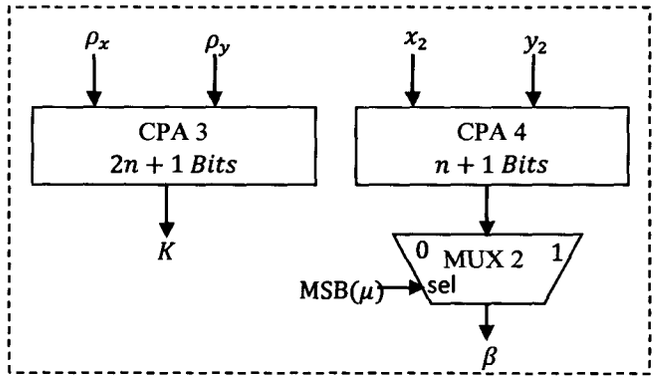

Figure 3.2: Overflow detection unit

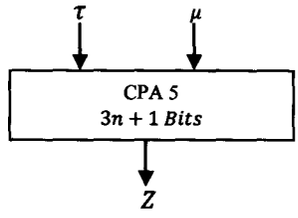

Figure 3.3: Correction Unit

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

3.1.4 Numerical Illustrations

This subsection looks at numerical examples with the proposed scheme.

Checking overflow in the sum of 49 and 21 using RNS moduli set {3, 4, 5}

$$49 = (1,1,4)_{RNS\langle 3|4|5\rangle} = (01,01,100)_{RNS\langle 11|100|101\rangle}$$

$$21 = (0,1,1)_{RNS\langle 3|4|5\rangle} = (00,01,001)_{RNS\langle 11|100|101\rangle}$$

$$= \big((01,01,100) + (00,01,001)\big)_{RNS\langle 11|100|101\rangle}$$

$$= (01,10,000)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(01,10,000)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 10. Whilst the sum of the decimal numbers 49 and 21 is 70 which is obvious of overflow occurring.

Checking for RNS overflow using the proposed algorithm

$$\rho_x = 12 = 01100$$

$$\rho_y = 5 = 00101$$

$$K = \rho_x + \rho_y = 01100 + 00101 = 10001$$

$$\mu = 01 + 01 = 010, \quad \beta = 0$$

Since the MSB of K is "1", the scheme will detect that overflow has occurred.

From (3.25),

$$Z = 1000100 + 0000010 = 1000110 = (70)_{\text{decimal}}$$

Checking overflow in the sum of 28 and 32 using RNS moduli set {3, 4, 5}

$$28 = (1,0,3)_{RNS\langle 3|4|5\rangle} = (01,00,011)_{RNS\langle 11|100|101\rangle}$$

$$32 = (2,0,2)_{RNS\langle 3|4|5\rangle} = (10,00,010)_{RNS\langle 11|100|101\rangle}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$= \left((01,00,011) + (10,00,010)\right)_{RNS\langle 11|100|101\rangle}$$

$$= (00,00,000)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(00,00,000)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 0. Whilst the sum of the decimal numbers 28 and 32 is 60, a clear sign of overflow occurring.

Checking for RNS overflow using the proposed algorithm

$$\rho_x = 7 = 00111$$

$$\rho_y = 8 = 01000$$

$$K = \rho_x + \rho_y = 00111 + 01000 = 01111$$

$$\mu = 00 + 00 = 000, \quad \beta = 0.$$

Even though, the MSB of K is not "1", all other bits are "1" therefore the scheme will detect that overflow has occurred.

From (3.25),

$$Z = 0111100 + 0000000 = 0111100 = (60)_{\text{decimal}}$$

Checking overflow in the sum of 10 and 11 using RNS moduli set {3, 4, 5}

$$10 = (1,2,0)_{RNS\langle 3|4|5\rangle} = (01,10,000)_{RNS\langle 11|100|101\rangle}$$

$$11 = (2,3,1)_{RNS\langle 3|4|5\rangle} = (10,11,001)_{RNS\langle 11|100|101\rangle}$$

$$= \left((01,10,000) + (10,11,001)\right)_{RNS\langle 11|100|101\rangle}$$

$$= (00,01,001)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(00,01,001)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 21 which is the correct result of 10 + 11.

Checking for RNS overflow using the proposed algorithm

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$\rho_x = 2 = 00010$$

$$\rho_y = 2 = 00010$$

$$K = 00010 + 00010 = 00100$$

$$\mu = 10 + 11 = 101, \ \ \beta = 1.$$

After processing, the scheme will obviously detect no overflow since it is only $K_{2n-2} = 1$. And from (3.25),

$$Z = 0010000 + 0000101 = 0010101 = (21)_{\text{decimal}}$$

3.1.5 Performance Evaluation

In order to evaluate the performance of the proposed overflow detection scheme, it is compared with similar best known state of the art schemes. Theoretical analysis from Table 3.1 shows that the proposed scheme has less delay and complexity without compromising on accuracy compared to that in (Younes & Steffan, 2013) which has a correction component. The proposed scheme is also faster than the scheme in (Siewobr & Gbolagade, 2014b). Even though, the hardware complexity of the proposed scheme is higher than that in (Siewobr & Gbolagade, 2014b), the proposed scheme uses three comparators and a single AND gate whilst the scheme (Siewobr & Gbolagade, 2014b) uses six comparators and three AND gates which are not included in the comparison. It is also clear from the AD^2 analysis that the proposed scheme is very efficient than the previous schemes. The correction part is not included in the evaluation just as it is not included in (Younes & Steffan, 2013) for fairness. In any case, the accurate result is a $(3n + 1)$ bit sum (Z) in (3.8) therefore, a $(3n + 1)$ bit adder is designed to cater for the addition.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

Table 3.1: Area and Delay Comparison for Proposed Scheme1

Scheme	**Area(Δ_{FA})**	**Delay(D_{FA})**	AD^2
Siewobr & Gbolagade, (2014b)	$11n + 6$	$22n + 12$	$5324n^3 + 81712n^2 + 4752n + 864$
Younes & Steffan, (2013)	$37n + 18$	$16n + \log n + 13$	$9472n^3 + 20000n^2 + 13661n + 3042$
Proposed	$19n + 2$	$6n + 3$	$684n^3 + 756n^2 + 243n + 18$

Table 3.2: Area and Delay analysis for various values of n

	AREA			**DELAY**			**AD^2**		
n	***S&G (2014)***	***Y&S (2013)***	***Proposed***	***S&G (2014)***	***Y&S (2013)***	***Proposed***	***S&G (2014)***	***Y&S (2013)***	***Proposed***
1	17	55	21	34	29	9	92652	33880	1701
2	28	92	40	56	45.3	15	379808	161550	9000
3	39	129	59	78	61.48	21	894276	442884	26019
4	50	166	78	100	77.6	27	1668000	934714	56862
5	61	203	97	122	93.7	33	2732924	1693872	105633
6	72	240	116	144	109.78	39	4120992	2777190	176436
7	83	277	135	166	125.85	45	5864148	4241500	273375
8	94	314	154	188	141.9	51	7994336	6143634	400554
9	105	351	173	210	157.95	57	10543500	8540424	562077
10	116	388	192	232	174	63	13543584	11488702	762048
Total	**665**	**2215**	**1065**	**1330**	**1016.56**	**360**	**47834220**	**36458350**	**2373705**

S&G: Siewobr & Gbolagade; Y&S: Younes & Steffan

Table 3.2 shows a detail analysis of the complexities and delay of the proposed scheme with similar state-of-the-art scheme by taking different values of n (in this case values of n from 1 to 10). The values obtained from this table is used to plot the graphs in Figures

www.udsspace.uds.edu.gh

3.4 to 3.6 in order to give a clear pictorial view of the performance of the proposed scheme. An error analysis from the table shows that the proposed scheme is about 95% more efficient in terms of the AD^2 values than the scheme by (Younes & Steffan, 2013) and over 96% than the scheme by (Siewobr & Gbolagade, 2014b) for the chosen values of n. The graph in Figure 3.6 depicts this clearly.

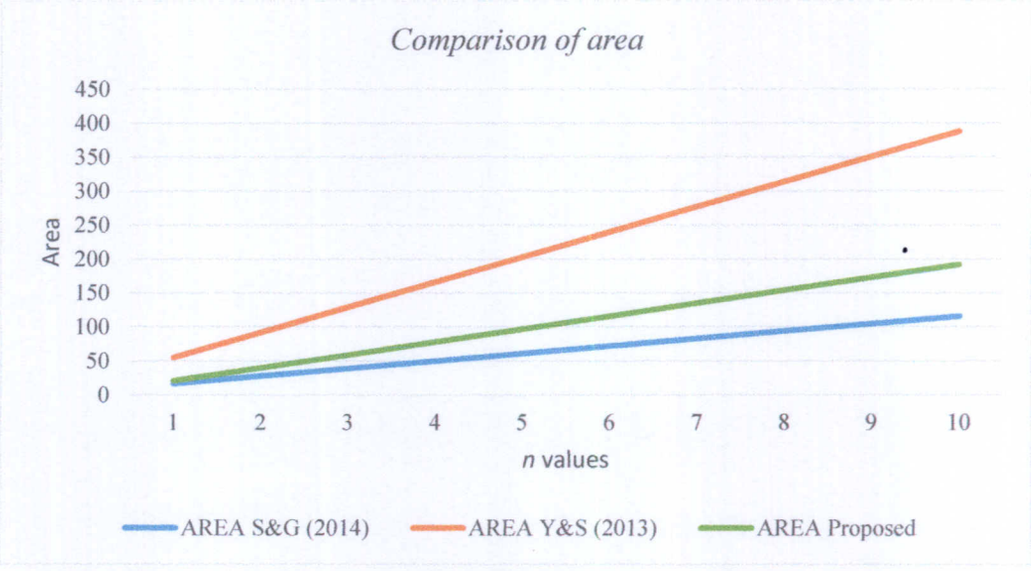

Figure 3.4: Graph of area comparison of proposed scheme1 with other schemes

From Figure 3.4, it clear that (Siewobr & Gbolagade, 2014b) requires the lesser resources and the Proposed Scheme also requires less than the scheme by (Younes & Steffan, 2013). Figure 3.5 is the graph of the delay comparison of the various schemes; it shows that the Proposed Scheme is much faster than the compared schemes.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

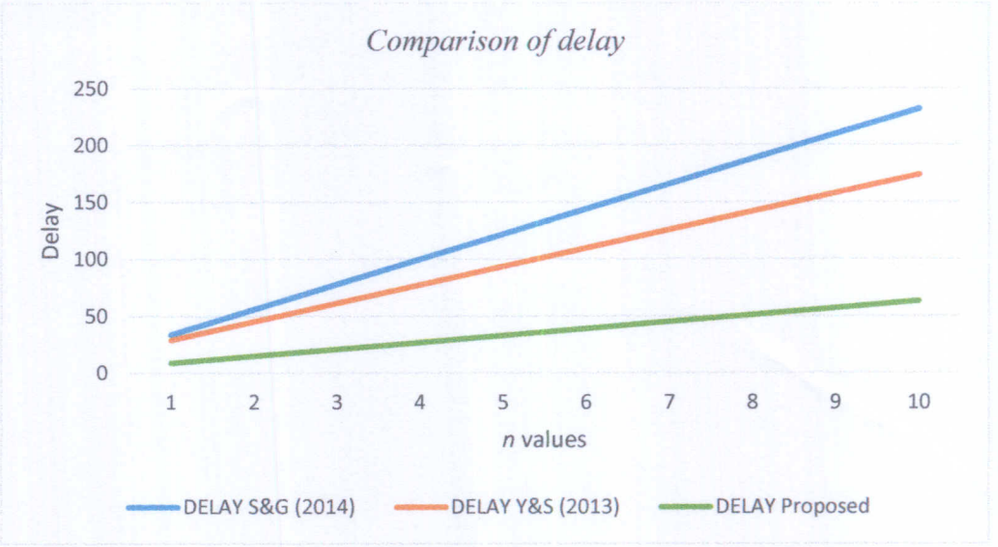

Figure 3.5: Graph of delay comparison of proposed scheme1 with other schemes

Figure 3.6 shows the efficiency comparison of the various schemes. As shown in this figure, the Proposed Scheme is the most efficient by plotting the AD^2 values of the schemes against the chosen values of n.

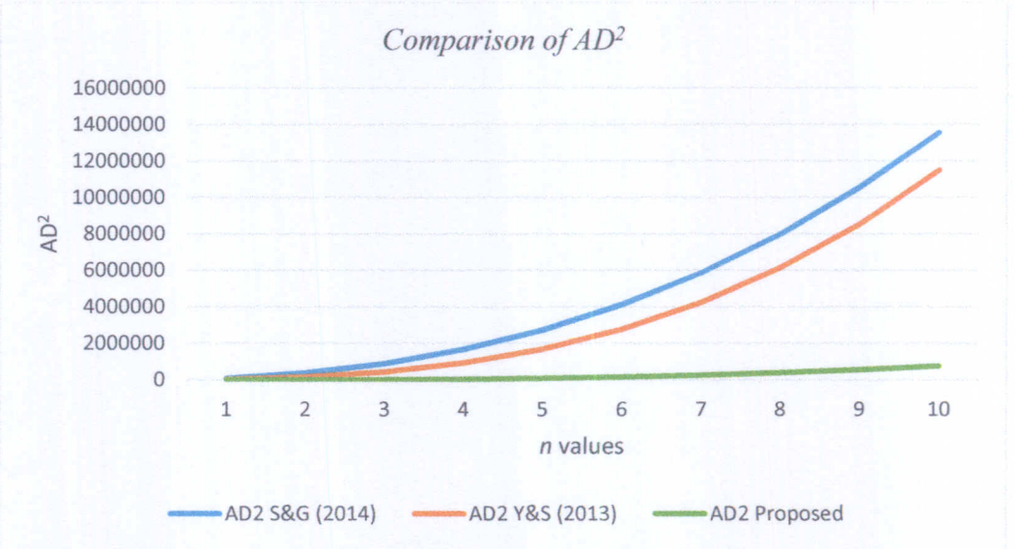

Figure 3. 6: Graph of AD^2 comparison of proposed scheme1 with other schemes

www.udsspace.uds.edu.gh

In the next section, another method for detecting and correcting overflow during addition using Mixed Radix Digits (MRDs) by magnitude evaluation will be presented for the same moduli set.

3.2 RNS Additive Overflow Detection and Correction Scheme through Magnitude Evaluation using Mixed Radix Digits

In this section, a new technique for detecting and correcting overflow during the addition of two RNS numbers for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is presented; the technique evaluates the sign of an RNS number by performing a partial reverse conversion using the mixed radix conversion method. The sign of the addends is evaluated using only the MRDs which is then used to detect the occurrence of overflow during RNS addition.

3.2.1 Proposed Method

Given the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, where $m_1 = 2^n + 1$, $m_2 = 2^n$ and $m_3 = 2^n - 1$, then

$$M = 2^n(2^n + 1)(2^n - 1) \tag{3.31}$$

This implies

$$\begin{aligned} M/2 &= 2^{n-1}(2^n + 1)(2^n - 1) \\ &= (2^n + 1)(2^{2n-1} - 2^{n-1}) \end{aligned} \tag{3.32}$$

Lemma 3.2: Given the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, where $m_1 = 2^n + 1$, $m_2 = 2^n$ and $m_3 = 2^n - 1$ for every integer $n > 1$, the following hold true, (Siewobr & Gbolagade, 2014b):

$$|m_1^{-1}|_{m_2} = 1 \tag{3.33}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$|m_2^{-1}|_{m_3} = 1 \tag{3.34}$$

$$|m_1^{-1}|_{m_3} = 2^{n-1} \tag{3.35}$$

Proof: If it can be demonstrated that $\left|m_i^{-1} \times m_i\right|_{m_i} = 1$, then m_i^{-1} is the multiplicative inverse of m_i with respect to m_i. Thus;

For (3.33), $|(2^n + 1) \times 1|_{2^n} = |1 \times 1|_{2^n}$

$$= |1|_{2^n} = 1$$

Also for (3.34), $|(2^n) \times 1|_{2^n-1} = |1 \times 1|_{2^n-1}$

$$= |1|_{2^n-1} = 1$$

Finally for (3.35), $|(2^n + 1) \times (2^{n-1})|_{2^n-1} = |2 \times (2^{n-1})|_{2^n-1}$

$$= |1|_{2^n-1} = 1 \qquad \blacksquare$$

Therefore we can re-write (2.3) as;

$$e_1 = x_1$$

$$e_2 = |(x_2 - e_1)1|_{2^n} = |x_2 - x_1|_{2^n}$$

$$e_3 = |((x_3 - e_1)2^{n-1} - e_2)1|_{2^n-1}$$

$$= |(x_3 - e_1)2^{n-1} - e_2|_{2^n-1} \tag{3.36}$$

Theorem 3.3: For the given moduli set, any integer $X \geq M/2$ if and only if

$$e_3 = 2^n - 2^{n-1} \tag{3.37}$$

or

$$e_3 = 2^n - 2^{n-1} - 1 \; AND \; e_2 = 2^n - 2^{n-1} \tag{3.38}$$

for any $n > 1$.

Proof: If it can be shown that by substituting (3.37) and (3.38) into equation (2.2) that,

$X \geq (2^n + 1)(2^{2n-1} - 2^{n-1})$ then, it implies $X \geq M/2$.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Assume (3.37) is true, then

$$X = e_1 + (2^n + 1)e_2 + (2^n - 2^{n-1})2^n(2^n + 1)$$

$$= e_1 + (2^n + 1)e_2 + (2^n + 1)(2^{2n} - 2^{2n-1})$$

$$= e_1 + (2^n + 1)[e_2 + 2^{2n} - 2^{2n-1}] > (2^n + 1)(2^{2n-1} - 2^{n-1}), \forall\, n > 1$$

Also, assume (3.38) is true, then

$$X \geq e_1 + (2^n - 2^{n-1})(2^n + 1) + (2^n - 2^{n-1} - 1)2^n(2^n + 1)$$

$$= e_1 + (2^n + 1)[(2^n - 2^{n-1}) + (2^{2n} - 2^{2n-1} - 2^n)]$$

$$= e_1 + (2^n + 1)[2^{2n} - 2^{2n-1} - 2^{n-1}] \geq (2^n + 1)(2^{2n-1} - 2^{n-1}), \forall\, n > 1 \quad \blacksquare$$

Thus, from (3.37) and (3.38), it is possible to determine the sign (from (1.1)) of an RNS number X whether $X \geq M/2$ (for a negative number) or $X < M/2$ (for a positive number).

The proposed method uses comparison by computing the MRDs of each of the addends to determine which half of the RNS range it belongs rather than performing a full reverse conversion to determine this. To detect overflow during addition of two addends X and Y based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, a single bit that indicates the sign of that addend is defined. Now, based on this bit, three cases will then be considered:

(i) Overflow will definitely occur if both of the addends are equal to or greater than half of the dynamic range ($M/2$).

(ii) Overflow will not occur if both of the addends are less than $M/2$.

(iii) Overflow may or may not occur if only one of the addends is equal or greater than $M/2$ and will require further processing to determine whether overflow will occur or not.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Let the magnitude evaluation of the addends (X, Y) be represented by β, such that if $\beta = 1$ or $\beta = 0$ represents a positive number or a negative number respectively as shown in equation (3.39). The evaluation of the undetermined case in (iii) is also represented by a single bit λ in (3.40).

$$\beta = \begin{cases} 1 \ ; & e_3 = 2^n - 2^{n-1} \\ 1 \ ; & e_3 = 2^n - 2^{n-1} - 1 \ AND \ e_2 = 2^n - 2^{n-1} \\ 0 \ ; & otherwise \end{cases} \tag{3.39}$$

and

$$\lambda = \begin{cases} 1 \ ; & x_1 + y_1 \geq 2^n + 1 \\ 0 \ ; & otherwise \end{cases} \tag{3.40}$$

From (3.39), it is possible to get a number that is less than half of the DR and another number that is greater than or equal to half of the DR. A further processing from (3.40) determines whether overflow will occur or not in such a situation.

The proposed method will then detect overflow as follows;

$$overflow = \begin{cases} 0 \ ; & \beta_X + \beta_Y = 0 \\ 1 \ ; & \beta_X \cdot \beta_Y = 1 \\ \lambda \ ; & \beta_X \oplus \beta_Y = 1 \end{cases} \tag{3.41}$$

where $(+, \cdot, \oplus)$ refer to the logical operations (OR, AND, XOR), respectively. For clarity, '1' means overflow occurs whilst '0' means no overflow.

Correction unit

The sum Z will normally be less than one of the addends once overflow occurs (thus the representation of an illegitimate number as if it is legitimate within the DR), this does not give the accurate result of the addition operation anytime overflow occurs. Let Z be the

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

sum of the two addends. By substituting the individual MRDs for both addends (X and Y), Z can be obtained as follows;

$$Z = X + Y$$

$$= [e_1(X) + e_2(X)m_1 + e_3(X)m_1m_2] + [e_1(Y) + e_2(Y)m_1 + e_3(Y)m_1m_2]$$

$$= \big(e_1(X) + e_1(Y)\big) + \big(e_2(X) + e_2(Y)\big)m_1 + \big(e_3(X) + e_3(Y)\big)m_1m_2$$

by letting $\psi_i = e_i(X) + e_i(Y)$, we shall have

$$Z = \psi_1 + \psi_2 m_1 + \psi m_1 m_2 \tag{3.42}$$

Thus by adding the individual MRDs of the two addends, we obtain the sum Z according (2.2) without having to compute separately for its MRDs. The value of Z obtained from (3.42) is the correct result of the addition whether overflow occurs or not. In case of overflow occurrence, M should be shifted one bit to the left in order to accommodate the value.

3.2.2 Hardware Implementation

From equation (3.36), the MRD's e_1, e_2 and e_3 can be represented in binary as;

$$e_1 = \underbrace{e_{1,n}e_{1,n-1}\cdots e_{1,1}e_{1,0}}_{n+1} \tag{3.43}$$

$$e_2 = \underbrace{e_{2,n-1}e_{2,n-2}\cdots e_{2,1}e_{2,0}}_{n} \tag{3.44}$$

$$e_3 = \underbrace{e_{3,n-1}e_{3,n-2}\cdots e_{3,1}e_{3,0}}_{n} \tag{3.45}$$

Equations (3.43) to (3.44) can further be simplified as follows;

$$e_1 = x_1 = \underbrace{x_{1,n}x_{1,n-1}\cdots x_{1,1}x_{1,0}}_{n+1} \tag{3.46}$$

$$e_2 = |x_2 - x_1|_{2^n} = |x_2 + t_1|_{2^n}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$= \left| \underbrace{x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0}}_{n} + \underbrace{t_{1,n-1}t_{1,n-2}\dots t_{1,1}t_{1,0}}_{n} \right|_{2^n}$$

$$= \underbrace{e_{2,n-1}e_{2,n-2}\dots e_{2,1}e_{2,0}}_{n} \tag{3.47}$$

where,

$$t_1 = |-x_1|_{2^n} = \left| -\left(\underbrace{x_{1,n}x_{1,n-1}\dots x_{1,1}x_{1,0}}_{n+1} \right) \right|_{2^n}$$

$$= \left| -\left(\underbrace{x_{1,n-1}x_{1,n-2}\dots x_{1,1}x_{1,0}}_{n} \right) \right|_{2^n}$$

$$= \left| \underbrace{\bar{x}_{1,n-1}\bar{x}_{1,n-2}\dots \bar{x}_{1,1}\bar{x}_{1,0}}_{n} \right|_{2^n} \tag{3.48}$$

and

$$e_3 = |2^{n-1}x_3 - 2^{n-1}e_1 - e_2|_{2^n-1} = |t_2 + t_3 + t_4|_{2^n-1}$$

$$= \left| \underbrace{t_{2,n-1}\dots t_{2,1}t_{2,0}}_{n} + \underbrace{t_{3,n-1}\dots t_{3,1}t_{3,0}}_{n} + \underbrace{t_{4,n-1}\dots t_{4,1}t_{4,0}}_{n} \right|_{2^n-1}$$

$$= \underbrace{e_{3,n-1}e_{3,n-2}\dots e_{3,1}e_{3,0}}_{n} \tag{3.49}$$

where

$$t_2 = |2^{n-1}x_3|_{2^n-1} = \left| 2^{n-1} \underbrace{\left(x_{3,n-1}x_{3,n-2}\dots x_{3,1}x_{3,0} \right)}_{n} \right|_{2^n-1}$$

$$= \underbrace{x_{3,0}x_{3,n-1}\dots x_{3,2}x_{3,1}}_{n} \tag{3.50}$$

$$t_3 = |-2^{n-1}x_1|_{2^n-1} = \left| -2^{n-1} \underbrace{\left(x_{1,n}x_{1,n-1}\dots x_{1,1}x_{1,0} \right)}_{n+1} \right|_{2^n-1}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$= \left| -2^{n-1}\left(x_{1,n} \times 2^{n} + \underbrace{x_{1,n-1} \dots x_{1,1}x_{1,0}}_{n}\right) \right|_{2^n-1} \quad (3.51)$$

Since, x_1 is a number that is smaller than $2^n + 1$, two cases are considered for x_1. First, when x_1 is smaller than 2^n, and second, when x_1 is equal to 2^n (Molahosseini *et al.*, 2010).

If $x_{1,n} = 0$, we have

$$t_{31} = \left| -2^{n-1} \underbrace{\left(x_{1,n-1}x_{1,n-2} \dots x_{1,1}x_{1,0}\right)}_{n} \right|_{2^n-1}$$

$$= \underbrace{\bar{x}_{1,0}\bar{x}_{1,n-1} \dots \bar{x}_{1,2}\bar{x}_{1,1}}_{n} \quad (3.52)$$

Else if $x_{1,n} = 1$, the following binary vector can be obtained as

$$t_{32} = \left| -2^{n-1} \times 2^{n}(\underbrace{00 \dots 0}_{n-1} x_{1,n}) \right|_{2^n-1} = 0\underbrace{11 \dots 1}_{n-1} \quad (3.53)$$

Therefore, t_3 is calculated as

$$t_3 = \begin{cases} t_{31}, & \text{if } x_{1,n} = 0 \\ t_{32}, & \text{if } x_{1,n} = 1 \end{cases} \quad (3.54)$$

And finally,

$$t_4 = |-e_2|_{2^n-1} = \left| -\underbrace{\left(e_{2,n-1}e_{2,n-2} \dots e_{2,1}e_{2,0}\right)}_{n} \right|_{2^n-1}$$

$$= \underbrace{\bar{e}_{2,n-1}\bar{e}_{2,n-2} \dots \bar{e}_{2,1}\bar{e}_{2,0}}_{n} \quad (3.55)$$

Let γ and ω represent the MRDs of the two integers X and Y respectively. Thus from equations (3.46) to (3.48), we have

$$\psi_i = \gamma_i + \omega_i \quad (3.56)$$

which implies

$$\psi_1 = \gamma_1 + \omega_1$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$= \underbrace{\gamma_{1,n}\gamma_{1,n-1} \dots \gamma_{1,0}}_{n+1} + \overbrace{\omega_{1,n}\omega_{1,n-1} \dots \omega_{1,0}}^{n+1}$$

$$= \psi_{1,n}\psi_{1,n-1} \dots \psi_{1,0} \tag{3.57}$$

$$\psi_2 = \gamma_2 + \omega_2$$

$$= \underbrace{\gamma_{2,n-1}\gamma_{2,n-2} \dots \gamma_{2,0}}_{n} + \overbrace{\omega_{2,n-1} \dots \omega_{2,0}}^{n}$$

$$= \psi_{2,n-1}\psi_{2,n-2} \dots \psi_{2,0} \tag{3.58}$$

finally,

$$\psi_3 = \gamma_3 + \omega_3$$

$$= \underbrace{\gamma_{3,n-1}\gamma_{3,n-2} \dots \gamma_{3,0}}_{n} + \overbrace{\omega_{3,n-1} \dots \omega_{3,0}}^{n}$$

$$= \psi_{3,n-1}\psi_{3,n-2} \dots \psi_{3,0} \tag{3.59}$$

and so, Z is implemented as;

$$Z = z_1 + z_3 + z_4$$

$$= \underbrace{\overbrace{0 \dots 0}^{2n}\underbrace{z_{1,n} \dots z_{1,1}z_{1,0}}_{n+1} + 0\underbrace{z_{3,3n-1} \dots z_{3,1}z_{3,0}}_{3n} + \overbrace{0 \dots 0}^{n+1}\underbrace{z_{4,2n-1} \dots z_{4,1}z_{4,0}}_{2n}}_{3n+1} \tag{3.60}$$

where,

$$z_1 = \psi_1 \tag{3.61}$$

$$z_3 = z_2 + 2^{2n}\psi_3$$

$$= \underbrace{\psi_{3,n-1} \dots \psi_{3,1}\psi_{3,0}}_{n}\overbrace{00 \dots 0}^{2n} \bowtie \underbrace{z_{2,2n-1} \dots z_{2,1}z_{2,0}}_{2n}$$

$$= z_{3,3n-1}z_{3,3n-2} \dots z_{3,1}z_{3,0} \tag{3.62}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$z_2 = 2^n\psi_2 + \psi_2$$

$$= \underbrace{\psi_{2,n-1} \cdots \psi_{2,1}\psi_{2,0}}_{n} \overbrace{00 \ldots 0}^{n} \bowtie \underbrace{\psi_{2,n-1} \cdots \psi_{2,1}\psi_{2,0}}_{n}$$

$$= z_{2,2n-1}z_{2,2n-2} \cdots z_{2,1}z_{2,0} \tag{3.63}$$

and,

$$z_4 = 2^n\psi_3$$

$$= \underbrace{\psi_{3,n-1} \cdots \psi_{3,1}\psi_{3,0}}_{n} \overbrace{00 \ldots 0}^{n}$$

$$= z_{4,2n-1}z_{4,2n-2} \cdots z_{4,1}z_{4,0} \tag{3.64}$$

3.2.3 Hardware Realisation

The hardware realisation of the proposed scheme is divided into four parts and shown in Figures 3.7 – 3.10. First is the Partial Conversion Part (PCP) shown in Figure 3.7, which evaluates the MRDs based on (3.46), (3.47) and (3.49) with their parameters clearly defined according to (3.48) and (3.49) – (3.55). The PCP begins with an Operands Preparation Unit (OPU) which prepares the operands in (3.48), (3.50) and (3.54) by simply manipulating the routing of the bits of the residues. Also, an n-bit 2: 1 Multiplexer (MUX) is used for obtaining (3.54). ADD1 is an n-bit Carry Propagate Adder (CPA) and is used to compute (3.47), meanwhile (3.49) is obtained by using an $(n-1)$-bit CPA as ADD2 whose save (s_1) and carry (c_1) are then added using ADD3 which is also an $(n-1)$-bit CPA. These MRDs are used to determine the sign of the RNS number in Figure 3.8. Thus, the critical path for the PCP unit is made up of one (2^n) modulo adder and two (2^n-1) modulo adders.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Second, is the Magnitude Evaluation Part (MEP) shown in Figure 3.8 which evaluates whether an RNS number is positive or negative according to (3.39). The MEP uses one AND gate and an OR gate. These are both two input monotonic gates. Next, is the Overflow Detection Part (ODP) which compares the sign bits of the two addends by using an AND gate according to (3.41) which is then ORed with the evaluated bit of the undetermined case in (3.40) as shown in Figure 3.9. This is where the scheme detects the occurrence of overflow during the addition of two numbers.

Lastly, in Figure 3.10 is the Overflow Correction Part (OCP). The OCP evaluates the individual MRDs of the two addends separately to achieve the sum Z in (3.42). This is done using five adders; four regular CPAs and one carry save adder (CSA). This is computed according to (3.57) – (3.64). ADD4, ADD5 and ADD6 add separately the MRDs e_1, e_2 and e_3 respectively for the two addends. The result of ADD4 is of importance because it is used in evaluating the undetermined case in (3.40). z_2 is a result of concatenation as well as z_3 which do not require any hardware. ADD7 is a CSA which computes the result of z_1, z_2 and z_3 whose save (s_2) and carry (c_2) are added using ADD 8 which is a CPA in order to get accurate sum Z whether overflow occurs or not. The schematic diagrams for the proposed scheme is presented in figures 3.7, 3.8 3.9 and 3.10.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

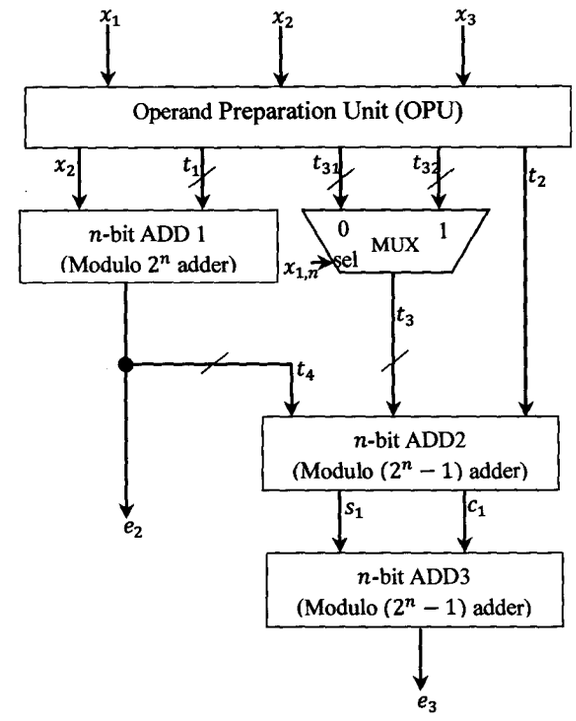

Figure 3.7: Partial Conversion Part (PCP)

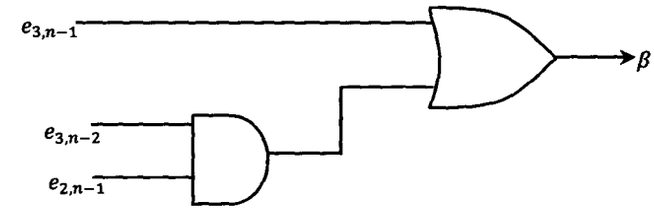

Figure 3.8: Magnitude Evaluation Part (MEP)

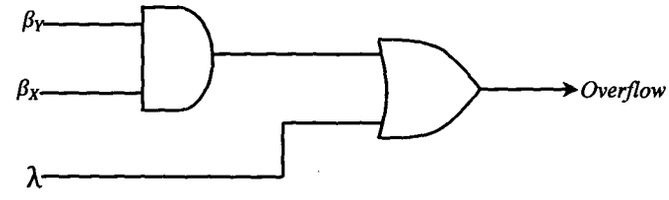

Figure 3.9: Overflow Detection Part (ODP)

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

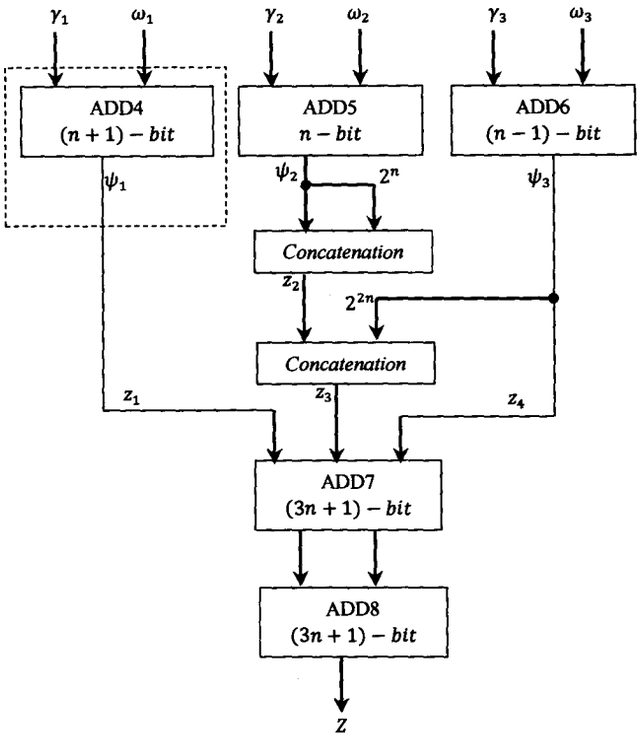

Figure 3.10: Overflow Correction Part (OCP)

The area (A) and time (D) requirements of the proposed scheme are estimated based on the unit-gate model as used in (Zimmermann, 1999 and Chang *et al.*, 2011) for fair comparison. In this model, each two-input monotonic gate such as AND, OR, NAND, NOR has area $A = 1$ and delay $D = 1$, each two-input gate XOR/XNOR has $A = D = 2$, The area and delay of an inverter is a negligible fraction of a unit, and it is thus assumed to require zero units of area and delay (Sousa, 2015). A 2:1 multiplexer has an area $A = 3$ and delay $D = 2$; A full adder has an area of seven gates and a delay of four gates but a CSA has a constant delay. Also, the adder requirements based on this model as presented in (Sousa, 2015) is adopted for the comparison since the adopted adders are similar to the adders used for the proposed scheme. The results state that an estimation modulo;

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$(2^n): A = 5n + \left(\frac{3}{2}\right) n \log_2 n, \qquad D = 2\log_2 n + 3$$

$$(2^n - 1): A = 12n + 3n(\log_2 n - 1), \quad D = 2\log_2 n + 3$$

Therefore, the hardware requirements of the scheme are as follows:

$$A_{PCP} = A_{ADD1} + A_{MUX} + A_{ADD2} + A_{ADD3} = 23n + \left(\frac{15}{2}\right)\log_2 n + 3$$

$$A_{MEP} = 2(A_{AND}) = 2$$

$$A_{ODP} = 2(A_{AND}) = 2$$

$$A_{OCP} = A_{ADD4} + A_{ADD5} + A_{ADD6} = 63n + 14$$

The estimated delay of the scheme will be as follows:

$$D_{PCP} = D_{ADD1} + D_{ADD2} + D_{ADD3} = 4\log_2 n + 5$$

$$D_{MEP} = 2(D_{AND}) = 2$$

$$D_{ODP} = 2(D_{AND}) = 2$$

$$D_{OCP} = D_{ADD4} + A_{ADD7} + A_{ADD8} = 9\ gates$$

Now, in order to make an effective comparison, the proposed scheme is divided into two: Proposed Scheme I for when the OCP is not included in the comparison and Proposed Scheme II for the OCP being included in the comparison. The delay of the OCP overrides the delay of the delays of the MEP and the ODP if Proposed Scheme II is consider since they will all be computed in parallel and the critical path in that case will be dictated by the OCP. The area for the PCP and the MEP is double for two numbers X and Y but this is not the case for the delay of the two numbers since they are computed in parallel. Thus, the total area and delay of the proposed schemes are:

$$A_{TOTAL(I)} = 2A_{PCP} + 2A_{MEP} + A_{ODP} = 46n + 15\log_2 n + 10$$

$$D_{TOTAL(I)} = D_{PCP} + D_{MEP} = 4\log_2 n + 7$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

That is when the overflow correction part is not included in the analysis of the scheme and,

$$A_{TOTAL(II)} = 2A_{PCP} + 2A_{MEP} + A_{ODP} + A_{OCP} = 109n + 15\log_2 n + 24$$

$$D_{TOTAL(II)} = D_{PCP} + D_{OCP} = 4\log_2 n + 16$$

For when the overflow correction part is included in estimating the requirements of the proposed scheme.

3.2.4 Numerical Illustrations

This subsection presents numerical illustrations of the proposed scheme.

Checking overflow in the sum of 49 and 21 using RNS moduli set {3, 4, 5}

$$X = 49 = (4,1,1)_{RNS\langle 5|4|3\rangle} = (100,01,01)_{RNS\langle 101|100|11\rangle}$$

$$Y = 21 = (1,1,0)_{RNS\langle 5|4|3\rangle} = (001,01,00)_{RNS\langle 101|100|11\rangle}$$

$$Z = \big((100,01,01) + (001,01,00)\big)_{RNS\langle 101|100|11\rangle}$$

$$= (000,10,01)_{RNS\langle 101|100|11\rangle}$$

RNS to decimal conversion of $(000,10,01)_{RNS\langle 101|100|11\rangle}$ will result in the decimal number 10. Whilst the sum of the decimal numbers 49 and 21 is 70 which is obvious of overflow occurring.

Checking for RNS overflow using the proposed technique

$$e_2(49) = |01 - 100|_{100} = 001$$

$$e_3(49) = |(01 - 01)(10) - 01|_{11} = |-01|_{11} = 10 = 2^2 - 2$$

Which implies, $\beta(49) = 1$ from (3.39)

Also, $e_2(21) = |01 + |-001|_{100}|_{100} = |01 + 11|_{100} = 000$

$$e_3(21) = |(11 - 01)(10) - 11|_{11} = |100 - 11|_{11} = 01 = 2^2 - 3$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

But $e_2(21)$ =000, which implies $\beta(21) = 0$ from (3.39)

Therefore, from (3.41) $overflow = \lambda$ and needs further processing.

Since $e_3(49) + e_3(21) = 10 + 01 = 11 > 2^2 - 2$, the scheme detects overflow occurring after processing.

Correction unit

$$Z = (100 + 001) + (01 + 00)(101) + (10 + 01)(101)(100)$$

$$= 1000110 = (70)_{decimal}$$

Checking overflow in the sum of 28 and 32 using RNS moduli set {3, 4, 5}

$$X = 28 = (3,0,1)_{RNS\langle 5|4|3\rangle} = (011,00,01)_{RNS\langle 101|100|11\rangle}$$

$$Y = 32 = (2,0,2)_{RNS\langle 5|4|3\rangle} = (010,00,10)_{RNS\langle 101|100|11\rangle}$$

$$Z = \big((011,00,01) + (010,00,10)\big)_{RNS\langle 101|100|11\rangle}$$

$$= (000,00,00)_{RNS\langle 101|100|11\rangle}$$

RNS to decimal conversion of $(000,00,000)_{RNS\langle 101|100|11\rangle}$ will result in the decimal number 0. Whilst the sum of the decimal numbers 28 and 32 is 60, a clear sign of overflow occurring.

Checking for RNS overflow using the proposed technique

$e_2(28) = |00 - 11|_{100} = |100 - 11|_{100} = 001 = 2^n - 3$

$e_3(28) = |(01 - 011)(10) - 01|_{11} = |10 - 01|_{11} = 01 = 2^2 - 3$

Which implies, $\beta(28) = 0$ since $e_2(28) = 001 < 2^n - 2$, from (3.39)

Also, $e_2(32) = |00 + |-010|_{100}|_{100} = |00 + 10|_{100} = 10 = 2^2 - 2$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$e_3(32) = |(10 - 010)(10) - 10|_{11} = |-10|_{11} = 01 = 2^2 - 3$

this implies $\beta(32) = 1$ since $e_3(32) = 01 = 2^2 - 3$ and $e_2(32) = 10 = 2^2 - 2$, from (3.39). Therefore, from (3.41) $overflow = \lambda$ and needs further processing.

Since $e_3(28) + e_3(32) = 01 + 01 = 10 = 2^2 - 2$, the scheme will detect that overflow occurred after processing.

Correction unit

$$Z = (011 + 010) + (01 + 10)(101) + (01 + 01)(101)(100)$$

$$= 111100 = (60)_{decimal}$$

Checking overflow in the sum of 10 and 11 using RNS moduli set {3, 4, 5}

$$X = 10 = (0,2,1)_{RNS\langle 5|4|3\rangle} = (000,10,01)_{RNS\langle 101|100|11\rangle}$$

$$Y = 11 = (1,3,2)_{RNS\langle 5|4|3\rangle} = (001,11,10)_{RNS\langle 101|100|11\rangle}$$

$$Z = \big((000,10,01) + (001,11,10)\big)_{RNS\langle 101|100|11\rangle}$$

$$= (001,01,00)_{RNS\langle 101|100|11\rangle}$$

RNS to decimal conversion of $(001,01,00)_{RNS\langle 101|100|11\rangle}$ will result in the decimal number 21 which is correct result of 10 + 11.

Checking for RNS overflow using the proposed algorithm

$e_2(10) = |10 - 000|_{100} = 10 = 2^2 - 2$

$e_3(10) = |(01 - 000)(10) - 10|_{11} = |10 - 10|_{11} = 00 < 2^2 - 3$

Which implies, $\beta(10) = 0$ since $e_3(10) = 00 < 2^2 - 3$, from (3.39)

Also, $e_2(11) = |11 + |-001|_{100}|_{100} = |11 + 11|_{100} = 10 = 2^2 - 2$

$e_3(11) = |(10 - 001)(10) - 10|_{11} = |10 - 10|_{11} = 00 < 2^2 - 3$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

this implies, $\beta(11) = 0$ since $e_3(10) = 00 < 2^2 - 3$, from (3.39).

ThFrom (3.41) $overflow = 0$, which implies no overflow has occurred according to the proposed scheme after processing.

Correction unit

$$Z = (000 + 001) + (10 + 10)(101) + (00 + 00)(101)(100)$$

$$= 010101 = (21)_{decimal}$$

3.2.5 Performance Evaluation

The performance of the proposed scheme is compared to schemes in (Younes & Steffan, 2013) and (Rouhifar *et al.*, 2011); the scheme in (Rouhifar *et al.*, 2011) does not contain a correction unit; the scheme by (Younes & Steffan, 2013) has a correction unit but is not included in the comparison. And so both schemes do not have the correction component in the comparison. Table 3.3 shows the analysis of the proposed scheme with that of similar state-of-the art schemes.

Table 3.3: Area, Delay comparison for scheme2

Scheme	***Area***	***Delay***
Rouhifar *et al.*, (2011)	$76n + (33/2)\,n\log_2 n$	$6\log_2 n + 23$
Younes & Steffan, (2013)	$37n + +18$	$16n + \log_2 n + 13$
Proposed Scheme I	$46n + 15\log_2 n + 10$	$4\log_2 n + 7$
Proposed Scheme II	$109n + 15\log_2 n + 24$	$4\log_2 n + 16$

As shown in Table 3.3, the proposed scheme for detecting overflow (in Proposed Scheme I) in the given moduli set is very cheap in terms of hardware resources and faster than the

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

scheme by (Rouhifar *et al.*, 2011) but requires a little hardware resources than the scheme by (Younes & Steffan, 2013) even though the Proposed Scheme I is completely faster than it. However, the complete proposed scheme (Proposed Scheme II) for detecting and correcting overflow requires more hardware resources than the other compared schemes but faster than both schemes by (Rouhifar *et al.*, 2011) and (Younes & Steffan, 2013). Clearly, Proposed Scheme I completely outperforms the similar state-of-the-art scheme by (Rouhifar *et al.*, 2011) for detecting overflow, but the trust of this work is to detect and correct overflow anytime it occurs; in so doing it has made tremendous gains in speed as shown in Table 3.3.

Table 3.4 shows a detailed analysis of the complexities and delay of the proposed scheme with that of the similar state-of-the-art schemes.

Table 3.4: Area, Delay analysis for various values of n for schemes2

	AREA				DELAY			
n	***R. et al (2011)***	***Y&S (2013)***	***Proposed* I**	***Proposed* II**	***R. et al (2011)***	***Y&S (2013)***	***Proposed* I**	***Proposed* II**
1	76	57	56	133	23	29	7	16
2	185	96	117	257	29	46	11	20
4	436	174	224	490	35	79	15	24
8	1004	330	423	941	41	144	19	28
16	2272	642	806	1828	47	273	23	32
32	5072	1266	1557	3587	53	530	27	36
64	11200	2514	3044	7090	59	1043	31	40
128	24512	5010	6003	14081	65	2068	35	44
256	53248	10002	11906	28048	71	4117	39	48
512	114944	19986	23697	55967	77	8214	43	52
Total	**212949**	**40077**	**47833**	**112422**	**500**	**16543**	**250**	**340**

R. et al: Rouhifar et al; Y&S: Younes & Steffan

Table 3.4 reveals interesting results theoretically, from the analysis it is clear that the Proposed Scheme I requires less resources than what is required by (Rouhifar *et al.*, 2011). From the table, smaller values of n shows that Proposed Scheme II requires more resources

www.udsspace.uds.edu.gh

than that by (Rouhifar *et al.*, 2011) but drastically improves upon this requirements up to over 51% better than (Rouhifar *et al.*, 2011) for higher values of n (i.e $n > 4$), this is clearly shown in the graph in Figure 3.11. The analysis from the table also shows that whilst for smaller values of n (say $n = 1$), the Proposed Scheme I is better than the scheme by (Younes & Steffan, 2013) in terms of hardware resources, it tends to require up to about 18% resources more than that by (Younes & Steffan, 2013).

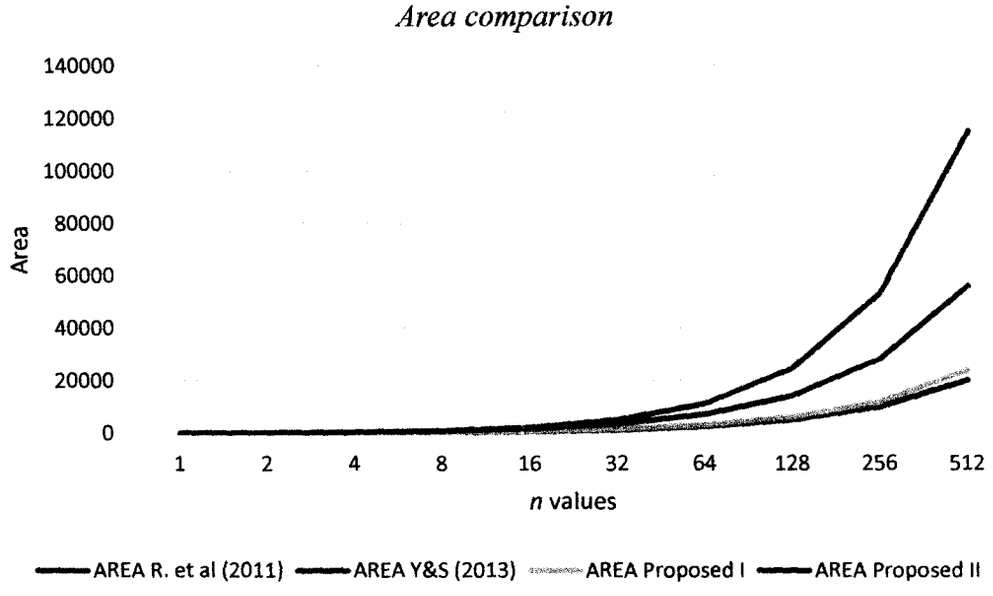

Figure 3.11: Graph of area analysis of proposed schemes2 with other schemes

From Figure 3.11, the scheme by (Rouhifar *et al.*, 2011) sharply increases for higher values of n followed by the Proposed Scheme II whilst the scheme by (Younes & Steffan, 2013) requires the lesser resources. Regarding the delay, the proposed schemes (Scheme I and Scheme II) completely outperforms both schemes by up to over 35% than the scheme by (Rouhifar *et al.*, 2011) and over 90% faster than the scheme by (Younes & Steffan, 2013) as shown in Table 3.4 and in Figure 3.12. It is worth noting that whiles the scheme by

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

(Younes & Steffan, 2013) is the best performer in terms of hardware resources, it tends to be the worst performer for speed and the percentage difference shows that the Proposed Scheme I is more efficient. Figure 3.12 is the graph analysis of the delay imposed by the various scheme. It is clear from the graph that the scheme in (Younes & Steffan, 2013) sharply increases with increasing values of n whiles the marginal increase of the rest of the schemes are minimal.

UNIVERSITY FOR DEVELOPMENT STUDIES

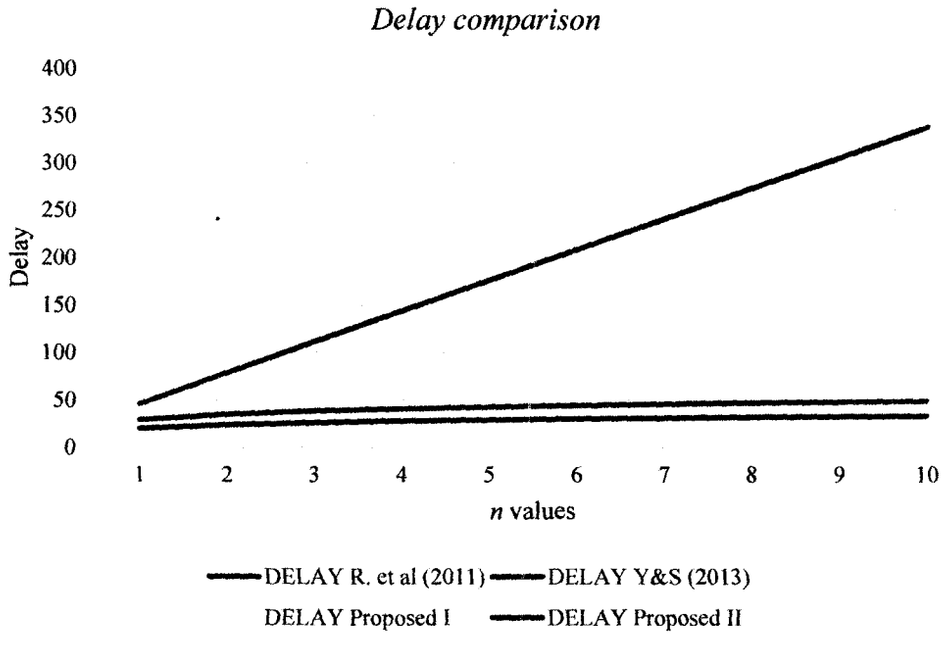

Figure 3.12: Bar graph of delay analysis of proposed schemes2 with other schemes

3.3 Conclusion

Detecting overflow in RNS arithmetic computations is very important but can be difficult, more so, if it has to be corrected. In this chapter, two schemes for detecting and correcting overflow during RNS addition for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ were presented; the first was a novel technique based on the CRT. The proposed technique did not require full

www.udsspace.uds.edu.gh

RNS-binary conversion. The proposed scheme was able to give the correct result for the sum of two numbers whether overflow occurred or not and the scheme was demonstrated theoretically to be very efficient than similar state of the art designs. The results of this method have been published (see Appendix). The second was an ingenious technique of detecting overflow by use of the MRC method through magnitude evaluation as well correcting the overflow when it occurs. This technique did not also require full reverse conversion but used the MRDs to evaluate the sign of a number to detect the occurrence of overflow. With this technique, the correct value of the sum of two numbers is always guaranteed whether overflow occurred or not. The scheme has been demonstrated theoretically be very fast than similar-state-of-the-art scheme but required little more hardware resources. However, the Proposed Scheme I, which is the one without the correction component completely outperformed the scheme by (Rouhifar *et al.*, 2011) in terms of both area and delay requirements. Also, results from Table 3.4 and Figure 3.11 showed that for higher values of n, the Proposed Scheme II also outperformed the scheme by (Rouhifar *et al.*, 2011).

The moduli set presented in this chapter and its variants for higher dynamic ranges are even moduli sets. Even dynamic range moduli sets are useful due to the fact that using the 2^n modulo greatly simplifies and reduces the delay and complexity of the residue arithmetic operations and the reverse conversion process. In the next chapter, another method for detecting and correcting overflow during addition in an odd dynamic range moduli set will be presented.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

CHAPTER FOUR

ADDITIVE OVERFLOW DETECTION AND CORRECTION FOR THE MODULI SET $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$

4.0 Introduction

The techniques presented in the previous chapter were for an even dynamic range. An additive overflow detection and correction scheme for the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$, which has odd dynamic range is presented in this chapter. The scheme uses a redundant modulus 2 by extending the dynamic range of the moduli set. This redundant modulus is then used to detect overflow during addition whenever it occurs by XORing the sum of the residues corresponding to the redundant modulus and the LSB of the result of summing the residues corresponding to two numbers in the original moduli set.

4.1 Proposed Method

Given the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$, (Bankas & Gbolagade, 2014) and (Molahosseini *et al.*, 2010), let $m_1 = 2^{2n+1}-1$, $m_2 = 2^n+1$ and $m_3 = 2^n-1$. Let $m_4 = 2$ be a redundant modulus by extending the original moduli set. In order to detect overflow in the given moduli set, this redundant modulus 2 is added so that the new set becomes $\{2^{2n+1}-1, 2^n+1, 2^n-1, 2\}$; but the dynamic range $M = (2^{2n+1}-1)(2^n+1)(2^n-1)$.

Theorem 4.1: Given the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$, where $m_1 = 2^{2n+1}-1$, $m_2 = 2^n+1$ and $m_3 = 2^n-1$, we have;

$$|m_1^{-1}|_{m_2} = 1 \tag{4.1}$$

$$|m_1^{-1}|_{m_3} = 1 \tag{4.2}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$|m_2^{-1}|_{m_3} = 2^{n-1} \tag{4.3}$$

Proof: If it can be shown that $|a \times b|_{m_i} = 1$, then $|b|_{m_i} = |a^{-1}|_{m_i}$ is the multiplicative inverse of a. Thus, for (4.1)

$$|(2^{2n+1} - 1) \times 1|_{2^n+1} = |(2(2^n)(2^n) - 1)|_{2^n+1}$$

$$= |(2(-1)(-1) - 1)|_{2^n+1}$$

$$= |(2 - 1)|_{2^n+1}$$

$$= |1|_{2^n+1} = 1$$

Hence 1 is the multiplicative inverse of m_1 with respect to m_2.

Similarly, for (4.2)

$$|(2^{2n+1} - 1) \times 1|_{2^n-1} = |(2(2^n)(2^n) - 1)|_{2^n-1}$$

$$= |(2 - 1)|_{2^n-1}$$

$$= |1|_{2^n-1} = 1$$

Hence 1 is the multiplicative inverse of m_1 with respect to m_3.

Also, for (4.3)

$$|(2^n + 1) \times 2^{n-1}|_{2^n-1} = |(2) \times 2^{n-1}|_{2^n-1}$$

$$= |2^n|_{2^n-1}$$

$$= |1|_{2^n-1} = 1$$

Hence 2^{n-1} is the multiplicative inverse of m_2 with respect to m_3.

Given the RNS numbers $X = (x_1, x_2, x_3, x_4)$ and $Y = (y_1, y_2, y_3, y_4)$.

Let the sum $Z = (z_1, z_2, z_3, z_4) = (x_i + y_i)$, $i = 1, 2, 3, 4$. Then two scenerios arise;

(i) If both addends have the same parity then $Z = (z_1, z_2, z_3)$ is even and $z_4 = 0$

(ii) If the addends have different parity then $Z = (z_1, z_2, z_3)$ is odd and $z_4 = 1$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Therefore, overflow occurs whenever $Z = (z_1, z_2, z_3)$ is odd and $z_4 = 0$ or $Z = (z_1, z_2, z_3)$ is even and $z_4 = 1$.

Thus the proposed method detects overflow as follows;

$$Overflow = \begin{cases} 1; & z_4 \ XOR \ LSB(Z) = 1 \\ 0, & Otherwise \end{cases} \tag{4.4}$$

Where $LSB(Z)$ is the least significant bit of the sum Z.

Next, a partial residue-binary conversion of the addends is done by computing their respective MRDs. The MRDs of one addend (say X) is done by substituting (4.1) – (4.3) into equation (2.3) to simplify as follows;

$$\begin{aligned} e_1 &= x_1 \\ e_2 &= |(x_2 - e_1)1|_{2^n+1} \\ &= |x_2 - x_1|_{2^n+1} \\ e_3 &= |((x_3 - e_1) - e_2)2^{n-1}|_{2^n-1} \\ &= |2^{n-1}(x_3 - x_1) - 2^{n-1}e_2|_{2^n-1} \end{aligned} \tag{4.5}$$

Therefore, Z is obtained by adding the individual MRDs of the two addends as in equation (3.42). In case of overflow occurrence, the dynamic range should be shifted one bit to the left thus including the modulus 2 in order to legitimize the value of Z. The value of Z computed this way is the correct result whether overflow occurs or not.

4.2 Hardware Implementation

From equation (4.5), the MRDs can be represented in binary as;

$$e_1 = \underbrace{e_{1,2n}e_{1,2n-1}\cdots e_{1,1}e_{1,0}}_{2n+1} \tag{4.6}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$e_2 = \underbrace{e_{2,n}e_{2,n-1}\dots e_{2,1}e_{2,0}}_{n+1} \tag{4.7}$$

$$e_3 = \underbrace{e_{3,n-1}e_{3,n-2}\dots e_{3,1}e_{3,0}}_{n} \tag{4.8}$$

Equations (4.6) – (4.8) can further be simplified as follows;

$$e_1 = x_1 = \underbrace{x_{1,2n}x_{1,2n-1}\dots x_{1,1}x_{1,0}}_{2n+1} \tag{4.9}$$

$$e_2 = |x_2 + A|_{2^n+1} = \left|\underbrace{x_{2,n}x_{2,n-1}\dots x_{2,1}x_{2,0}}_{n+1} + \underbrace{A_nA_{n-1}\dots A_1A_0}_{n+1}\right|_{2^n+1}$$

$$= \underbrace{e_{2,n}e_{2,n-1}\dots e_{2,1}e_{2,0}}_{n+1} \tag{4.10}$$

where,

$$A = \left|-\underbrace{(x_{1,2n}x_{1,2n-1}\dots x_{1,1}x_{1,0})}_{2n+1}\right|_{2^n+1}$$

$$= \left|\underbrace{11\dots1\bar{x}_{1,2n}}_{n} + \underbrace{\bar{x}_{1,2n-1}\bar{x}_{1,2n-2}\dots\bar{x}_{1,n}}_{n} + \underbrace{\bar{x}_{1,n-1}\bar{x}_{1,n-2}\dots\bar{x}_{1,0}}_{n}\right|_{2^n+1} \tag{4.11}$$

Let $A_1 = \underbrace{11\dots1\bar{x}_{1,2n}}_{n}$, $A_2 = \underbrace{\bar{x}_{1,2n-1}\bar{x}_{1,2n-2}\dots\bar{x}_{1,n}}_{n}$ and $A_3 = \underbrace{\bar{x}_{1,2n-1}\bar{x}_{1,2n-2}\dots\bar{x}_{1,n}}_{n}$ (4.12)

Also,

$$e_3 = |2^{n-1}(x_3 - x_1) - 2^{n-1}e_2|_{2^n-1}$$

$$= \left|\underbrace{x_{3,0}x_{3,n-1}x_{3,n-2}\dots x_{3,1}}_{n} + \underbrace{B_{n-1}B_{n-1}\dots B_1B_0}_{n} + \underbrace{C_{n-1}C_{n-1}\dots C_1C_0}_{n}\right|_{2^n-1}$$

$$= \underbrace{e_{3,n-1}e_{3,n-2}\dots e_{3,1}e_{3,0}}_{n} \tag{4.13}$$

where,

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$B = |-2^{n-1}x_1|_{2^n-1} = \left|-2^{n-1}\underbrace{(x_{1,2n}x_{1,2n-1}\dots x_{1,1}x_{1,0})}_{2n+1}\right|_{2^n-1}$$

$$= \left|-2^{n-1}\underbrace{(00\dots 0x_{1,2n})}_{n} -2^{n-1}\underbrace{(x_{1,2n-1}\dots x_{1,n})}_{n} -2^{n-1}\underbrace{(x_{1,n-1}\dots x_{1,0})}_{n}\right|_{2^n-1}$$

$$= \left|\underbrace{\bar{x}_{1,2n}11\dots 1}_{n} + \underbrace{\bar{x}_{1,n}\bar{x}_{1,2n-1}\dots\bar{x}_{1,n+1}}_{n} + \underbrace{\bar{x}_{1,0}\bar{x}_{1,n-1}\dots\bar{x}_{1,1}}_{n}\right|_{2^n-1} \quad (4.14)$$

Let $B_1 = \underbrace{x_{3,0}x_{3,n-1}x_{3,n-2}\dots x_{3,1}}_{n}$, $B_2 = \underbrace{\bar{x}_{1,2n}11\dots 1}_{n}$, $B_3 = \underbrace{\bar{x}_{1,n}\bar{x}_{1,2n-1}\dots\bar{x}_{1,n+1}}_{n}$ and

$$B_3 = \underbrace{\bar{x}_{1,0}\bar{x}_{1,n-1}\dots\bar{x}_{1,1}}_{n} \quad (4.15)$$

Finally,

$$C = |-2^{n-1}e_2|_{2^n-1} = \left|-2^{n-1}\underbrace{(e_{2,n}e_{2,n-1}\dots e_{2,1}e_{1,0})}_{n+1}\right|_{2^n-1}$$

$$= \left|-2^{n-1}\left(e_{2,n}\times 2^n + \underbrace{e_{2,n-1}\dots e_{2,1}e_{1,0}}_{n}\right)\right|_{2^n-1} \quad (4.16)$$

Since, e_2 is a number that is smaller than $2^n + 1$, two cases can be considered. First, when e_2 is smaller than 2^n, and second, when e_2 is equal to 2^n (Molahosseini *et al.*, 2010).

If $e_{2,n} = 0$, we have

$$C_1 = \left|-2^{n-1}\underbrace{(e_{2,n-1}e_{2,n-2}\dots e_{2,1}e_{1,0})}_{n}\right|_{2^n-1}$$

$$= \underbrace{\bar{e}_{2,0}\bar{e}_{2,n-1}\dots\bar{e}_{2,2}\bar{e}_{2,1}}_{n} \quad (4.17)$$

Else if $e_{2,n} = 1$, the following binary vector can be obtained as

$$C_2 = \left|-2^{n-1}\times 2^n(\underbrace{00\dots 0}_{n-1}e_{2,n})\right|_{2^n-1} = 0\,\underbrace{11\dots 1}_{n-1} \quad (4.18)$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Therefore, t_3 is calculated as

$$C = \begin{cases} C_1, & \text{if } e_{2,n} = 0 \\ C_2, & \text{if } e_{2,n} = 1 \end{cases} \tag{4.19}$$

Let γ and ω represent the MRDs of the two integers X and Y respectively. Then from equations (4.9), (4.10) and (4.13), we have

$$\psi_i = \gamma_i + \omega_i \tag{4.20}$$

which implies

$$\begin{aligned} \psi_1 &= \gamma_1 + \omega_1 \\ &= \underbrace{\gamma_{1,n}\gamma_{1,n-1} \cdots \gamma_{1,1}\gamma_{1,0}}_{2n+1} + \overbrace{\omega_{1,n}\omega_{1,n-1} \cdots \omega_{2,1}\omega_{1,0}}^{2n+1} \\ &= \psi_{1,2n}\psi_{1,n-1} \cdots \psi_{1,1}\psi_{1,0} \end{aligned} \tag{4.21}$$

$$\begin{aligned} \psi_2 &= \gamma_2 + \omega_2 \\ &= \underbrace{\gamma_{2,n}\gamma_{2,n-1} \cdots \gamma_{2,1}\gamma_{2,0}}_{n+1} + \overbrace{\omega_{2,n}\omega_{2,n-1} \cdots \omega_{2,1}\omega_{2,0}}^{n+1} \\ &= \psi_{2,n}\psi_{2,n-1} \cdots \psi_{2,1}\psi_{2,0} \end{aligned} \tag{4.22}$$

finally,

$$\begin{aligned} \psi_3 &= \gamma_3 + \omega_3 \\ &= \underbrace{\gamma_{3,n-1}\gamma_{3,n-2} \cdots \gamma_{3,1}\gamma_{3,0}}_{n} + \overbrace{\omega_{3,n-1}\omega_{3,n-2} \cdots \omega_{3,1}\omega_{3,0}}^{n} \\ &= \psi_{3,n-1}\psi_{3,n-2} \cdots \psi_{3,1}\psi_{3,0} \end{aligned} \tag{4.23}$$

and so, Z is implemented as;

$$Z = t_2 + t_3 + t_4 + t_5 + t_6$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$= \underbrace{t_{2,4n} \dots t_{1,0}}_{4n+1} + \overbrace{0 \dots 0}^{n-1} \underbrace{t_{3,2n+1} \dots t_{3,0}}_{3n+2} + \overbrace{0 \dots 0}^{3n+1} \underbrace{t_{4,n-1} \dots z_{4,1} z_{4,0}}_{n} + \overbrace{0 \dots 0}^{3n} \underbrace{t_{5,n} \dots t_{5,0}}_{n+1} + \overbrace{0 \dots 0}^{2n+1} \underbrace{t_{6,n} \dots t_{6,0}}_{2n} \quad (4.24)$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{4n+1}$$

where,

$$t_2 = 2^{3n+1}\psi_3 + t_1$$

$$= \underbrace{\psi_{3,n-1}\psi_{3,n-2} \dots \psi_{3,1}\psi_{3,0}}_{n} \overbrace{00 \dots 0}^{3n+1} \bowtie \underbrace{t_{1,3n} \dots t_{1,1}t_{1,0}}_{3n+1}$$

$$= t_{2,4n} \dots t_{2,1}t_{2,0} \quad (4.25)$$

and,

$$t_1 = 2^{2n+1}\psi_3 + \psi_1$$

$$= \underbrace{\psi_{3,n-1} \dots \psi_{3,1}\psi_{3,0}}_{n} \overbrace{00 \dots 0}^{2n+1} \bowtie \underbrace{\psi_{1,2n} \dots \psi_{1,1}\psi_{1,0}}_{2n+1}$$

$$= t_{1,3n}t_{1,3n-1} \dots t_{1,1}t_{1,0} \quad (4.46)$$

$$t_3 = 2^{2n+1}\psi_2$$

$$= \underbrace{\psi_{2,n}\psi_{2,n-1} \dots \psi_{2,1}\psi_{2,0}}_{n+1} \overbrace{00 \dots 0}^{2n+1}$$

$$= t_{3,3n+1}t_{3,3n} \dots t_{3,1}t_{3,0} \quad (4.27)$$

$$t_4 = -\psi_3$$

$$= \underbrace{\bar{\psi}_{3,n-1}\bar{\psi}_{3,n-2} \dots \bar{\psi}_{3,1}\bar{\psi}_{3,0}}_{n}$$

$$= t_{4,n-1}t_{4,n-2} \dots t_{4,1}t_{4,0} \quad (4.28)$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$t_5 = -\psi_3$$

$$= \underbrace{\bar{\psi}_{2,n}\bar{\psi}_{2,n-1} \cdots \bar{\psi}_{2,1}\bar{\psi}_{2,0}}_{n}$$

$$= t_{5,n}t_{5,n-1} \cdots t_{5,1}t_{5,0} \tag{4.29}$$

finally,

$$t_6 = -2^n\psi_3$$

$$= \underbrace{\bar{\psi}_{3,n-1}\bar{\psi}_{3,n-2} \cdots \bar{\psi}_{3,1}\bar{\psi}_{3,0}}_{n}\overbrace{11 \ldots 1}^{n}$$

$$= t_{6,2n-1}t_{6,2n-1} \cdots t_{6,1}t_{6,0} \tag{4.30}$$

4.2.1 Hardware Realisation

The hardware architecture of the proposed scheme is first realised by computing the MRDs of the two addends X and Y according to (4.10) and (4.13) which parameters are defined in (4.11), (4.12), (4.14), (4.15) and (4.19). This MRDs are e_2 and e_3, and e_1 in (4.2) which is equivalent to x_1. Figure 4.1 shows the unit for computing the MRDs of one addend X and repeated for the other addend Y. Figure 4.1 consists of a two level Carry Save Adder (CSA) tree for computing e_2 and another three level CSA tree for computing e_3 whose sum and carry are added using two separate CPAs each. This unit is called here Partial Reverse Converter (PRC) as a component of the proposed scheme. The PRC starts with an Operands Preparation Unit (OPU 1), which prepares the operands in (4.12) and (4.15) by simply manipulating the routing of the bits of the residues. The operands in (4.12) added with CSA 1 at a first level and at a second level includes x_2 in CSA 3 which sum and carry are added using CPA 1 to get e_2. A multiplexer is used to determine (4.19) by either choosing (4.17) or (4.18) depending on the MSB of e_2. The value from (4.19) and the

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

operands in (4.15) are then added using the three level CSA tree in CSA 2, CSA 4 and CSA 5 and finally propagated with CPA 2 in order to get e_3. These MRDs are useful in computing the sum of the addends Z by the Reverse Converter (RC) in Figure (4.2). The respective MRDs of the addends are summed according to (4.21) – (4.23) and computed according to CPA 3, CPA 4 and CPA 5. These and other four adders made up the architecture for the reverse converter for the sum Z in Figure 4.2; after an operand preparation (4.24) is computed by a three level carry save tree in CSA 2, CSA3 and CSA 4 in a cascading manner whose sum and carry are then added using CPA6 in order to get Z which the correct result of the addition operation whether overflow occurs or not. Finally, overflow is detected by XORing the LSB(Z) with $|Z|_2 = z_4$ according to (4.4) and shown in Figure 4.3.

The hardware complexities and delay (time required for processing) of the proposed scheme are estimated as follows;

The area (A) and delay (D) of the PRC are:

$$A_{PRC} = A_{CSA1} + A_{CSA2} + A_{CSA3} + A_{CSA4} + A_{CSA5} + A_{CPA1} + A_{CPA2}$$

$$= (n+1)\Delta_{FA} + n\Delta_{FA} + (n+1)\Delta_{FA} + n\Delta_{FA} + n\Delta_{FA} + n\Delta_{FA} + n\Delta_{FA}$$

$$= (7n+2)\Delta_{FA}$$

$$D_{PRC} = D_{CSA1} + D_{CSA3} + D_{CPA1} + D_{CSA5} + D_{CPA1} + D_{CPA2}$$

$$= D_{FA} + D_{FA} + 2nD_{FA} + D_{FA} + 2nD_{FA}$$

$$= (4n+3)D_{FA}$$

The area requirement and delay imposed by the RC are:

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$A_{RC} = A_{CPA3} + A_{CPA4} + A_{CPA5} + A_{CSA2} + A_{CSA3} + A_{CSA4} + A_{CPA6}$$

$$= (2n + 1)\Delta_{FA} + (n + 1)\Delta_{FA} + (n - 1)\Delta_{FA} + 3(4n + 1)\Delta_{FA} + (4n + 1)\Delta_{FA}$$

$$= (20n + 5)\Delta_{FA}$$

$$D_{PRC} = D_{CPA3} + D_{CSA2} + D_{CSA3} + D_{CSA4} + D_{CPA6}$$

$$= (4n + 2)D_{FA} + 3D_{FA} + (8n + 2)D_{FA} = (12n + 7)D_{FA}$$

The ODU is a two input XOR gate and requires a unit of gate each for the area and delay. Equations (4.25) and (4.26) are realised by merely joining (concatenating) bits since the sum of a and $2^n b$ is computed as b concatenation a if a is an n-bit number (Bankas & Gbolagade, 2013b), hence does not require any hardware or impose a delay. Also, the area for two addends will be double in the case of the PRC but the same delay.

Therefore, the total area requirements and delay of the proposed scheme are:

$$A_{TOTAL} = 2A_{PRC} + A_{RC} + A_{ODU} = (14n + 4)\Delta_{FA} + (20n + 5)\Delta_{FA} + \Delta_{FA}$$

$$= (34n + 10)\Delta_{FA}$$

$$D_{TOTAL} = D_{PRC} + D_{RC} + D_{ODU} = (4n + 3)D_{FA} + (12n + 7)D_{FA} + D_{FA}$$

$$= (16n + 11)D_{FA}$$

The schematic diagrams of the proposed scheme are shown in figures 4.1, 4.2 and 4.3.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

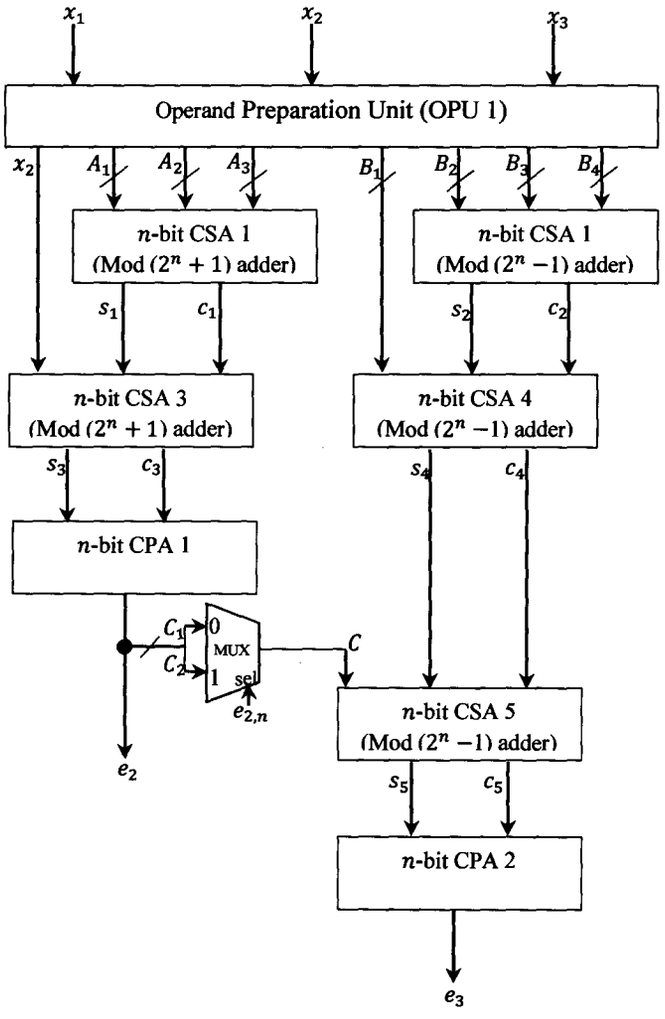

Figure 4.1: Partial Reverse Converter (PRC)

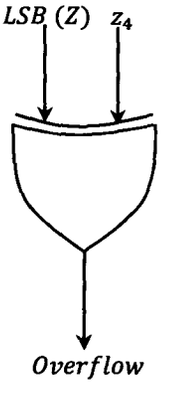

Figure 4.2: Overflow Detection Unit (ODU)

www.udsspace.uds.edu.gh

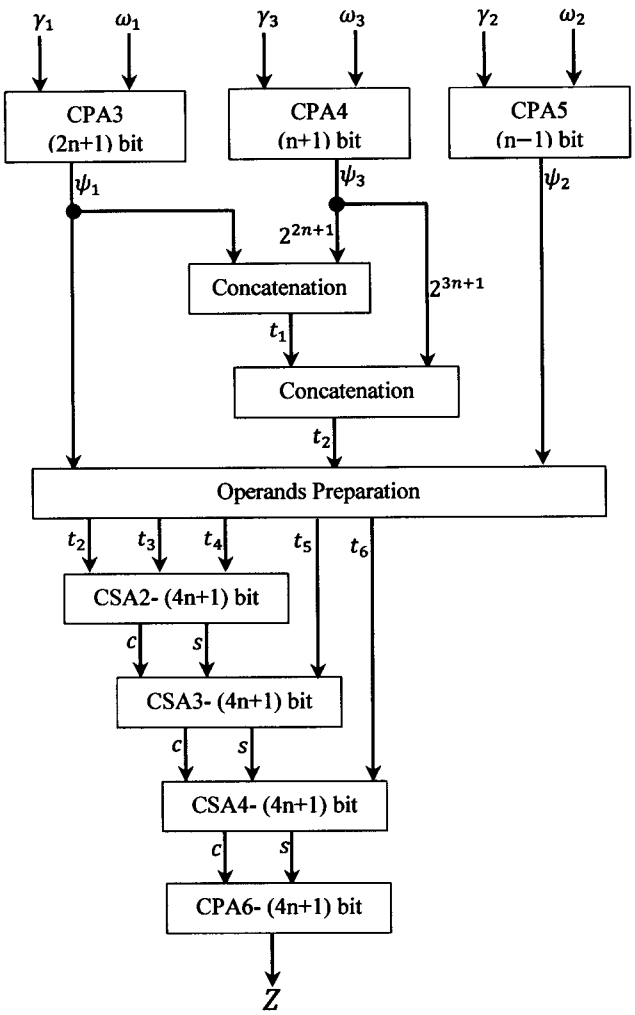

Figure 4.3: Reverse Converter (RC)

4.3 Numerical Illustrations

This section presents numerical illustrations of the proposed scheme.

Checking overflow in the sum of 225 and 275 using RNS moduli set $\{31, 5, 3, 2\}$.

Legitimate range (DR) = 465. Let;

$$X = 225 = (8, 0, 0, \mathbf{1})_{RNS(31|5|3|2)} = (01000, 000, 00, \mathbf{1})_{RNS(31|5|3|2)}$$

$$Y = 275 = (27, 0, 2, \mathbf{1})_{RNS(31|5|3|2)} = (11011, 000, 10, \mathbf{1})_{RNS(31|5|3|2)}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

$$Z = \left((01000, 000, 00, \mathbf{1}) + (11011, 000, 10, \mathbf{1})\right)_{RNS(31|5|3|2)}$$

$$= (00100, 000, 10, \mathbf{0})_{RNS(31|5|3|2)}$$

RNS to decimal conversion of $(00100, 000, 10)_{RNS(31|5|3)}$ results in decimal number 35. Meanwhile the sum of 225 and 275 is 500, a clear case of overflow occurrence.

Checking for RNS overflow using the proposed method

$Z = (00100, 000, 10, \mathbf{0})_{RNS(31|5|3|2)}$ implies $\boldsymbol{z_4} = \mathbf{0}$ and

$Z = (00100, 000, 10)_{RNS(31|5|3)} = 100011_{BINARY}$ which implies $LSB(Z) = 1$

Therefore, $LSB(Z) XOR\ \boldsymbol{z_4} = 1\ XOR\ \mathbf{0} = \mathbf{1}$. Thus overflow has occurred according to the proposed method since both numbers have the same parity.

Correction part

The correct value of Z is RNS to decimal conversion of $(00100, 000, 10, \mathbf{0})_{RNS(31|5|3|2)}$ which results in decimal number 500.

Checking overflow in the sum of 225 and 322 using RNS moduli set $\{31, 5, 3, 2\}$.
Legitimate range (DR) = 465. Let;

$$X = 225 = (8, 0, 0, \mathbf{1})_{RNS(31|5|3|2)} = (01000, 000, 00, \mathbf{1})_{RNS(31|5|3|2)}$$

$$Y = 322 = (12, 2, 1, \mathbf{0})_{RNS(31|5|3|2)} = (01100, 010, 01, \mathbf{0})_{RNS(31|5|3|2)}$$

$$Z = \left((01000, 000, 00, \mathbf{1}) + (01100, 010, 01, \mathbf{0})\right)_{RNS(31|5|3|2)}$$

$$= (10100, 010, 01, \mathbf{1})_{RNS(31|5|3|2)}$$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

RNS to decimal conversion of $(10100, 010, 01)_{RNS(31|5|3)}$ results in decimal number 82. Meanwhile the sum of 225 and 322 is 547, a clear case of overflow occurrence.

Checking for RNS overflow using the proposed method

$Z = (10100, 010, 01, \mathbf{1})_{RNS(31|5|3|\mathbf{2})}$ implies $\boldsymbol{z_4} = \mathbf{1}$ and

$Z = (10100, 010, 01)_{RNS(31|5|3)} = 1010010_{BINARY}$ which implies $LSB(Z) = 0$

Therefore, $LSB(Z) XOR\ \boldsymbol{z_4} = 0\ XOR\ \mathbf{1} = \mathbf{1}$. Thus overflow has occurred according to the proposed method since both numbers have different parity.

Correction part

The correct value of Z is RNS to decimal conversion of $(10100, 010, 01, \mathbf{1})_{RNS(31|5|3|\mathbf{2})}$ which results in decimal number 547.

Checking overflow in the sum of 225 and 35 using RNS moduli set $\{31, 5, 3, 2\}$.
Legitimate range (DR) = 465. Let;

$$X = 225 = (8, 0, 0, \mathbf{1})_{RNS(31|5|3|\mathbf{2})} = (01000, 000, 00, \mathbf{1})_{RNS(31|5|3|\mathbf{2})}$$

$$Y = 35 = (4, 1, 1, \mathbf{1})_{RNS(31|5|3|\mathbf{2})} = (00100, 001, 01, \mathbf{1})_{RNS(31|5|3|\mathbf{2})}$$

$$Z = \big((01000, 000, 00, \mathbf{1}) + (00100, 001, 01, \mathbf{1})\big)_{RNS(31|5|3|\mathbf{2})}$$

$$= (001100, 001, 01, \mathbf{0})_{RNS(31|5|3|\mathbf{2})}$$

RNS to decimal conversion of $(001100, 001, 01)_{RNS(31|5|3)}$ results in decimal number 260 which is the correct result of summing 225 and 35. In this case overflow has not occurred.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Checking for RNS overflow using the proposed method

$Z = (001100, 001, 01, \mathbf{0})_{RNS(31|5|3|2)}$ implies $\mathbf{z_4 = 0}$ and

$Z = (001100, 001, 01)_{RNS(31|5|3)} = 100000100_{BINARY}$ which implies $LSB(Z) = 0$

Therefore, $LSB(Z) XOR\ \mathbf{z_4} = 0\ XOR\ \mathbf{0} = \mathbf{0}$. Thus overflow has not occurred according to the proposed method.

Since overflow has not occurred, there will not be any need for the correction unit.

4.4 Performance Evaluation

The performance of the proposed scheme is compared to similar schemes of equal dynamic range reverse converter as well as the scheme by (Askarzadeh *et al.*, 2009) that have odd dynamic range. The complexities that are considered here for the analysis are a Full Adder (FA), a Half Adder (converted to FA) and a two input XOR gate. It is also worth noting that the complexities (area) as presented in (Mohan, 2008) are for a single number (say X) and so would have to be doubled in order to take care of two numbers (say X and Y) for the reverse conversion process. Table 4.1 presents the complexities and delay by the various schemes for the purpose of comparison.

Table 4.1: Area and Delay analysis of proposed scheme3 with similar schemes of equal DR

Scheme	AREA	DELAY
(Mohan, 2008)	$(28n + (5n^2/2) + 12)\Delta_{FA}$	$(18n + 23)D_{FA}$
(Askarzadeh et al., 2009)	$(48n + 21)\Delta_{FA}$	$(16n + 15)D_{FA}$
Proposed Scheme	$(34n + 10)\Delta_{FA}$	$(16n + 11)D_{FA}$

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

From table 4.1, it is obvious that the proposed scheme is better than (Askarzadeh *et al.*, 2009) in terms of the area complexities even though the delay is almost the same, but the proposed scheme has a correction component. Also, the proposed scheme performs better than the scheme by (Mohan, 2008) for higher values of n, in both area and delay. A detailed analysis is presented in Table 4.2 taking some values of n, thus it gives a clearer picture of the results presented in Table 4.1 for some values of n.

Table 4.2: Area, Delay analysis for various values of n for scheme3

	AREA			**DELAY**		
n	***Mohan, (2008)***	***Askarzadeh et al., (2009)***	***Proposed***	***Mohan, (2008)***	***Askarzadeh et al., (2009)***	***Proposed***
1	42.5	69	44	41	31	27
2	78	117	78	59	47	43
4	164	213	146	95	79	75
8	396	405	282	167	143	139
16	1100	789	554	311	271	267
32	3468	1557	1098	599	527	523
64	12044	3093	2186	1175	1039	1035
128	44556	6165	4362	2327	2063	2059
256	171020	12309	8714	4631	4111	4107
512	669708	24597	17418	9239	8207	8203
Total	**902577**	**49314**	**34882**	**18644**	**16518**	**16478**

Table 4.2 shows detailed analysis of the area and delay comparison of scheme3 for various values of n with similar-state of the art schemes. The results from Table 4.2 are used to plot the graphs in Figure 4.4 and Figure 4.5; Figure 4.4 is a graph of area comparison of the various schemes. It shows that the proposed scheme requires the lesser area than the other schemes. Figure 4.5 also presents the graph of the delay comparison of the compared schemes which shows however that the proposed scheme and the scheme by (Askarzadeh *et al.*, 2009) have almost the same speed but performs better than the scheme by (Mohan, 2008).

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

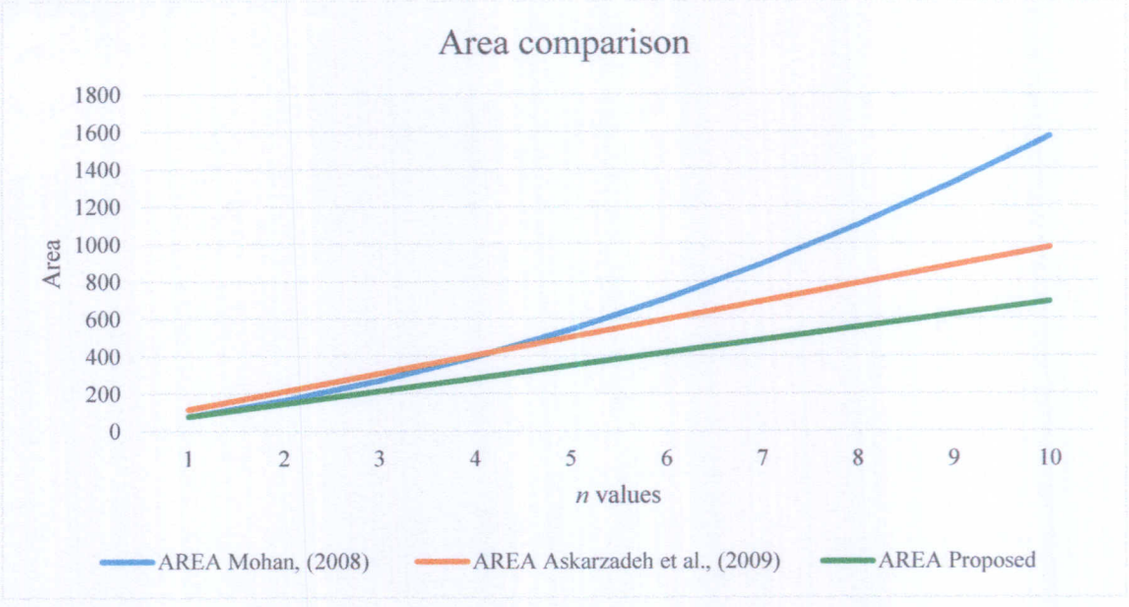

Figure 4.4: Graph of area analysis of proposed scheme3 with other schemes

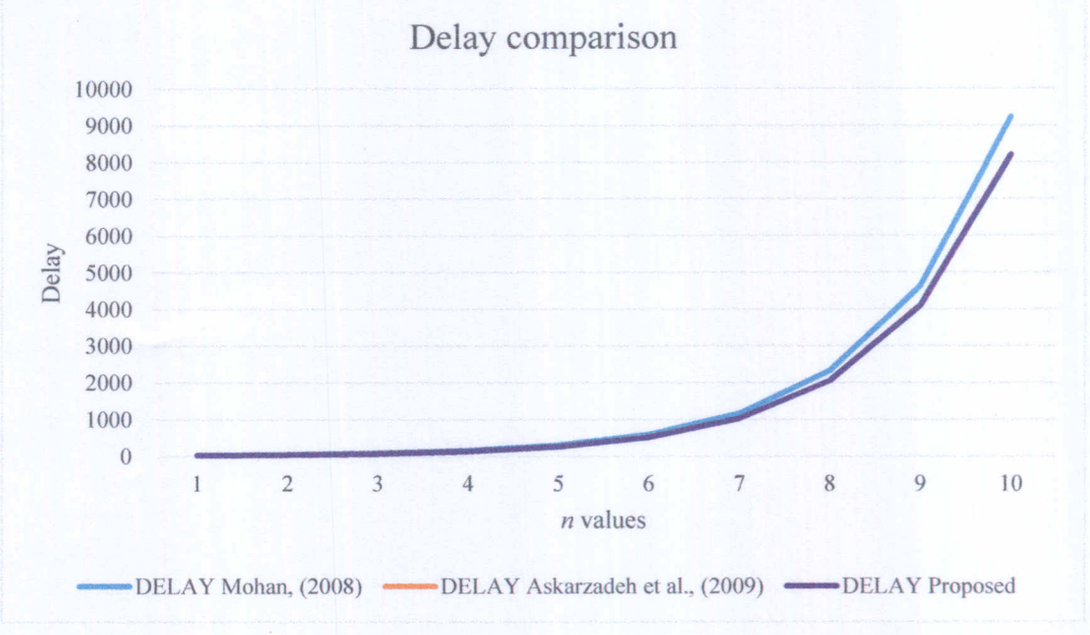

Figure 4.5: Graph of delay analysis of proposed scheme3 with other schemes

www.udsspace.uds.edu.gh

4.5 Conclusion

In this chapter, an additive overflow detection and correction scheme for the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$ was presented. The scheme used a redundant modulus 2 by extending the dynamic range of the moduli set. This redundant modulus was then used to detect overflow during addition whenever it occurred by XORing the sum of the residues corresponding to the redundant modulus and the LSB of the result of summing the residues corresponding to two numbers in the original moduli set. The proposed scheme has been demonstrated theoretically to be an efficient scheme by comparing it to previous similar works.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

CHAPTER FIVE

SUMMARY AND FUTURE RESEARCH

5.0 Introduction

In the last two previous chapters, the proposed schemes were presented with their performance evaluation. This chapter concludes the thesis by presenting a summary of the thesis as well as projections into the future of possible research areas of the researcher.

5.1 Summary

Overflow detection is one of the fundamental challenges that hinders the widespread usage of the RNS. If the RNS is used in applications that require fast computations such as DSPs and Transforms, then there should be a guarantee that values (numbers) used for such computations are always accurate. The occurrence of overflow will result in wrong representation of numbers as if they are correct in the RNS system. This makes the issue of overflow detection one of great importance. Detecting overflow in RNS representation is necessary but not sufficient if RNS is to have that universal usage in computing as a general purpose processor. There is the need to be able to correct the overflow in the system whenever it occurs, thus overflow correction is very crucial. The goal of this thesis was to devise techniques of detecting and correcting overflow in RNS arithmetic computations and by so doing three techniques were developed: Two efficient schemes (scheme1 and scheme2) for RNS overflow detection and correction for a generalised moduli set $\{2^{\alpha n}-1, 2^{\alpha n}, 2^{\alpha n}+1\}$, $\alpha = 1, 2, \ldots$ was proposed. Then by taking $\alpha = 1$, the algorithms were applied to the popular moduli set $\{2^n-1, 2^n, 2^n+1\}$ for

www.udsspace.uds.edu.gh

implementation and comparison but α can be increased for higher dynamic ranges. The first was a novel technique based on the CRT, which did not require full RNS-binary conversion. This proposed scheme prevented the representation of illegitimate numbers in the RNS system as if they were legitimate numbers thus correcting overflow. The second technique was based on the MRC method by evaluating the magnitude of numbers and then using that to detect and correct overflow if it occurred. This technique did not also require full reverse conversion but used the MRDs to evaluate the sign of a number to detect the occurrence of overflow. With this technique, the correct value of the sum of two numbers is guaranteed whether overflow occurred or not. Both schemes were demonstrated theoretically to be very efficient than similar state-of-the art schemes. These were presented in chapter three.

The other scheme (scheme3) though based on the MRC for the reverse conversion was applied on the moduli set $\{2^{2n+1}-1, 2^n+1, 2^n-1\}$ with a redundant modulus 2. The redundant modulus was used to devise a simple way of detecting overflow in an RNS system during addition by XORing the LSB of the sum of the addition with the residue corresponding to the redundant modulus. The reverse conversion process of the sum will yield the correct result whether overflow occurred or not. This technique has also been demonstrated theoretically to be competing favourably with similar previous works. This scheme was presented in chapter four. Chapter one introduced the concept of number systems: weighted and non-weighted, and the RNS capitalising on the speed limitation imposed on the weighted number systems to gain prominence in research due to its inherent futures desirable in applications requiring faster computations with addition and multiplication being dominant. A bit of history of the RNS is presented in chapter two with

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

the review of some previous works on overflow detection. The motivation of the thesis came from the review of such literature.

5.2 Recommendations/Future Research Works

Currently the widely used RNS arithmetic computations are addition and multiplication since subtraction and to some extend division are the reverse process (variants) of either addition or multiplication. The goal of this work was to develop schemes to detect and correct overflow during RNS arithmetic computations, but the schemes presented are for detecting and correcting overflow during RNS addition leaving out multiplication. It is the hope of the researcher to in future look at ways of developing schemes to handle multiplication by designing algorithms that when implemented will not require a lot of hardware complexities and will also be fast.

The schemes that are presented are also implemented theoretically due to the unavailability of laboratories and lack of Field Programmable Gate Array (FPGA) boards. It will be interesting to practically implement the proposed schemes to see their efficiency. In this regard, the researcher would in future works, tries to implement the proposed schemes and any other schemes practically subject to the availability of laboratories or in their absence funds to be able to procure the FPGA boards to do the implementation.

Lastly, the RNS processor would only become a general purpose processor if some of the challenges such as overflow detection and correction, reverse conversion, sign and magnitude comparison and some difficult arithmetic operations such division, scaling and root function are well addressed. In future, the researcher would continue to address some

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

of these challenges of the RNS system by working on other efficient schemes to handle such challenges as done in this thesis for overflow detection and correction.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

UNIVERSITY FOR DEVELOPMENT STUDIES

REFERENCES

Askarzadeh, M., Hosseinzadeh, M., & Navi, K. (2009). A New Approach to Overflow Detection in Moduli Set $\{2^n\text{-}3, 2^n\text{-}1, 2^n + 1, 2^n + 3\}$. In *Second International Conference on Computer and Electrical Engineering, (ICCEE '09)*. (1), pp. 439–442.

Bankas, E. K. (2013). *Efficient Residue to Binary Converters for some Powers of Two Moduli Sets* (PhD Thesis). University for Development Studies, Tamale, Ghana.

Bankas, E. K., & Gbolagade, K. A. (2012). A Speed Efficient RNS to Binary Converter for the Moduli Set $\{2^n, 2^n + 1, 2^n\text{-}1\}$. *Journal of Computing*, *4*(5), pp. 83–88.

Bankas, E. K., & Gbolagade, K. A. (2013a). An Effective New CRT Based Converter for a Novel Modulli Set $\{2^{2n+1}\text{-}1, 2^{2n}, 2^{2n}\text{-}1\}$. In *24th International Conference on Application-specific Systems, Architecture and Processors (ASAP2013)* (pp. 142–146). Washington, USA.

Bankas, E. K., & Gbolagade, K. A. (2013b). A New Efficient FPGA Design of Residue-To-Binary Converter. *International Journal of VLSI Design & Communication Systems (VLSICS)*, *4*(6).

Bankas, E. K., & Gbolagade, K. A. (2013c). A Residue to Binary Converter for a Balanced Moduli set $\{2^{2n+1}\text{-}1, 2^{2n}, 2^{2n}\text{-}1\}$. In *International Joint Conference on Computing* (pp. 21–216).

Bankas, E. K., & Gbolagade, K. A. (2014). A New Efficient RNS Reverse Converter for the 4-Moduli Set $\{2^n, 2^n + 1, 2^n\text{-}1, 2^{2n+1}\text{-}1\}$. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *8*(2), pp. 318–322.

www.udsspace.uds.edu.gh

Bhardwaj, M., Srikanthan, T., & Clarke, C. T. (1999). A reverse converter for the 4 moduli super set $\{2^n-1, 2^n, 2^n + 1, 2^{n+1} + 1\}$. *IEEE Conference on Computer Arithmetic*.

Chang, C. H., Low, J., & Yung, S. (2011). Simple, fast, and exact RNS scaler for the three-moduli set $\{2^n-1, 2^n, 2^n + 1\}$. *IEEE Transaction on Circuits and Systems I: Regular Papers*, *58*(11), pp. 2686–2697.

Daabo, M. I. & Gbolagade, K. A. (2014). An Overflow Detection Scheme with a Reverse Converter for the Moduli Set $\{2^n-1, 2^n, 2^n + 1\}$. *Journal of Emerging Trends in Computing and Information Sciences*, *5*(12), pp. 931–935.

Daabo, M. I. (2015). *Overflow Detection Schemes for Residue Number System Architecture* (PhD Thesis). University for Development Studies, Tamale, Ghana.

Daabo, M. I., & Gbolagade, K. A. (2012). RNS Overflow Detection Scheme for the Moduli set {M - 1, M}. *Journal of Computing*, *4*(8), pp. 39–44.

Debnath, R. C., & Pucknell, D. A. (1978). On multiplicative overflow detection in residue number system. *Electronics Letters*, *14*(5), pp. 129–130. http://doi.org/10.1049/el:19780088

Dugdale, M. (1992). VLSI implementation of residue adders based on binary adders. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *39*(5), pp. 325–329. http://doi.org/10.1109/82.142036

Etzel, M., & Jenkins, W. (1980). Redundant residue number systems for error detection and correction in digital filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *28*(5), pp. 538–545. http://doi.org/10.1109/TASSP.1980.1163442

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Flynn, M. J., & Huang, P. (2005). Microprocessor Design: thoughts on the road ahead. *IEEE Transactions on Microprocessors*, *25*(3), pp. 16–31.

Gbolagade, K. A. (2010). *Effective Reverse Conversion in Residue Number System Processors* (PhD Thesis). Delft University of Technology, The Netherlands.

Gbolagade, K. A., Chaves, R., Sousa, L., & Cotofana, S. D. (2010). An improved reverse converter for $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set. *IEEE International Symposium on Circuits and Systems (ISCAS 2010)*, pp. 2103–2106.

Gbolagade, K. A., & Cotofana, S. D. (2008). MRC Technique for RNS to Decimal Conversion for the moduli set $\{2n + 2, 2n + 1, 2n\}$ (pp. 318–321). *Presented at the 16th Annual Workshop on Circuits, Systems, and Signal Processing*, Veldhoven, The Netherlands.

Hosseinzadeh, M., Molahosseini, A. S., & Navi, K. (2009). A parallel Implementation of the Reverse Converter for the moduli set $\{2^n-1, 2^n, 2^{n-1}-1\}$. *World Academy of Science, Engineering and Technology*, *55(1)*, pp. 494–498.

Jaberipur, G., & Ahmadifar, H. (2014). A ROM-less reverse RNS converter for moduli set $\{2q \pm 1, 2q \pm 3\}$. *IET Computers Digital Techniques*, *8*(1), pp. 11–22. http://doi.org/10.1049/iet-cdt.2012.0148

Keir, Y. A., Cheney, P. W., & Tannenbaum, M. (1962). Division and Overflow Detection in Residue Number Systems. *IRE Transactions on Electronic Computers*, *EC-11*(4), pp. 501–507. http://doi.org/10.1109/TEC.1962.5219389

Kettani, H. (2006). On the Conversion Between Number Systems. *IEEE Transaction on Circuits and Systems-II*, *51*(11), pp. 1255–1258.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Lu, M. (2004). *Arithmetic and Logic in Computer Systems* (1st ed.). New Jersey, USA: A John Wiley & Sons, Inc., Publication.

Mohan, P. V. A. (2007). RNS-To Binary Converter for a New Three-Moduli set. *IEEE Transaction on Circuits and Systems-II, 54*(9), pp. 775–779.

Mohan, P. V. A. (2008). New reverse converters for the moduli set $\{2^n-3, 2^n-1, 2^n + 1, 2^n + 3\}$. *International Journal of Electronics and Communication, 62(1)*, pp. 643–658.

Molahosseini, A. S., Navi, K., Dadkhah, C., Kavehei, O., & Timarchi, S. (2010). Efficient Reverse Converter Designs for the New 4-Moduli sets $\{2^n-1, 2^n, 2^n + 1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n + 1, 2^n, 2^{2n} + 1\}$ Based on New CRTs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *57*(4), pp. 823–835.

Nannarelli, A., Re, M., & Cardarilli, G. C. (2001). Tradeoffs between residue number system and traditional FIR filters. In *in IEEE Int. Symp. Circuits Syst.* (pp. 305–308).

Narayanaswamy, N. (2010, December). *Optimisation of New Chinese REmainder Theorems Using Special Moduli Sets* (MSc Thesis). Louisiana State University, and Agricultural and Mechanical College, India.

Omondi, A., & Premkumar, B. (2007). *Residue Number Systems: Theory and Implementation* (2nd ed.). Published by Imperial College Press and Distributed by World Scientific Publishing Co. Retrieved from http://www.worldscientific.com/worldscibooks/10.1142/p523

Parhami, B. (2000). *Computer Arithmetic Algorithms and Hardware Designs* (1st ed.). New York: Oxford University Press.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Piestrak, S. J. (1995). A high-speed realization of a residue to binary number system converter. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, pp. 661–663.

Rouhifar, M., Hosseinzadeh, M., Bahanfar, S., & Teshnehlab, M. (2011). Fast Overflow Detection in Moduli set $\{2^n\text{-}1, 2^n, 2^n + 1\}$. *International Journal of Computer Science Issues*, 8*(3)*, pp. 407–414.

Shang, M., Jian Hao, H., Lin, Z., & Xiang, L. (2008). An efficient RNS parity checker for moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ and its applications. *Springer Journal of Science in China Series F: Information Science*, *51*(10), pp. 1563–1571.

Sheu, M., Lin, S. H., Chen, C., & Yang, S. W. (2004). An efficient VLSI design for a residue to binary converter for general balance moduli set $\{2^n\text{-}3, 2^n\text{-}1, 2^n + 1, 2^n + 3\}$. *IEEE Transactions on Circuits and Systems II*, *51*(3), pp. 152–155.

Siewobr, H., & Gbolagade, K. A. (2011). An Overflow Detection in Residue Number Systems Addition before Forward Conversion. *International Journal of Computational Intelligence and Information Security*, *2*(9), pp. 48–54.

Siewobr, H., & Gbolagade, K. A. (2014a). Residue-to-Binary Converter for the New Moduli Set $\{2^{(3n+2)/2}\text{-}1, 2^{2n}, 2^{(3n+2)/2} + 1\}$. *IOSR Journal of VLSI and Signal Processing*, *4*(1), pp. 44–49.

Siewobr, H., & Gbolagade, K. A. (2014b). RNS Overflow Detection by Operands Examination. *International Journal of Computer Applications*, *85*(18), pp. 1–5. http://doi.org/10.5120/14938-2906

Singh, N. (2008). An overview of Residue Number System. Presented at the National Seminar on Devices, Circuits & Communication, Mesra, Ranchi.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

Soderstrand, M. A., Jenkins, W. K., Jullien, G. G., & Taylor, F. J. (1986). *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. NJ, USA: IEEE press Piscataway.

Soderstrand, M. A., Jenkins, W. K., Jullien, G. G., & Taylor, F. J. (1986). *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. NJ, USA: IEEE press Piscataway.

Sousa, L. (2015). 2^n RNS Scalers for Extended 4-Moduli Sets. *IEEE Transactions on Computers*, (99), pp. 1–14. http://doi.org/10.1109/TC.2015.2401026

Theodore, L. H. (1989). Residue Addition Overflow Detection Processor. *Seatle, Wash Appl.*, (414276).

Younes, D. (2013). *Residue Number Based Building Blocks for Applicaitons in Digital Signal Processing* (PhD Thesis). Brno University of Technology, Czech Republic.

Younes, D., & Steffan, P. (2013). Universal Approaches for Overflow and Sign Detection in Residue Number System Based on $\{2^n - 1, 2^n, 2^n + 1\}$ (pp. 77–81). *Presented at the ICONS 2013, The Eighth International Conference on Systems*. Retrieved from http://www.thinkmind.org/index.php?view=article&articleid=icons_2013_4_20_20091

Zimmermann, R. (1999). Efficient VLSI implementation of modulo (2^n±1) addition and multiplication. *14th IEEE Symposium on Computer Arithmetic*, pp. 158 – 167.

UNIVERSITY FOR DEVELOPMENT STUDIES

www.udsspace.uds.edu.gh

APPENDIX

Publication from the thesis

Agbedemnab, P. A., & Bankas, E. K. (2015). A Novel RNS Overflow Detection and Correction Algorithm for the Moduli Set $\{2^n - 1, 2^n, 2^n + 1\}$. *International Journal of Computer Applications*, *110*(16), pp. 30–34. http://doi.org/10.5120/19403-0925

UNIVERSITY FOR DEVELOPMENT STUDIES