

UNIVERSITY FOR DEVELOPMENT STUDIES

AN EFFICIENT RNS SCALER FOR A CERTAIN MODULI SET

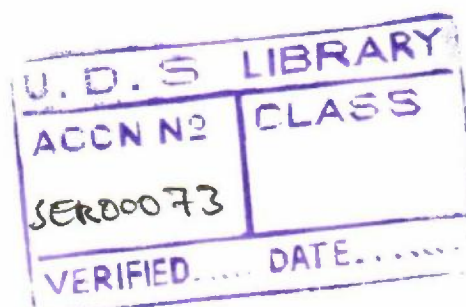
JIBREEL FUSEINI

UNIVERSITY FOR DEVELOPMENT STUDIES



Thesis submitted to the Department of Mathematics, Faculty of Mathematical Sciences,
University for Development Studies in partial fulfilment of the requirements for the award
of Master of Science Degree in Computational Mathematics

2011



UNIVERSITY FOR DEVELOPMENT STUDIES

AN EFFICIENT RNS SCALER FOR A CERTAIN MODULI SET

BY

JIBREEL FUSEINI (BSc. COMPUTER SCIENCE)

(UDS/MM/0003/09)



Thesis submitted to the Department of Mathematics, Faculty of Mathematical Sciences,
University for Development Studies in partial fulfilment of the requirements for the award of
Master of Science Degree in Computational Mathematics


November, 2011

DECLARATION

I hereby declare that this dissertation is the result my original work and that no part of it has been presented for another degree in this University or elsewhere. Related works by others which served as a source of knowledge has been duly referenced and acknowledged.

Candidate's Signature:  Date: 21-11-2011
Name: TIBERIEL FUSEINI

I hereby declare that the preparation and presentation of this dissertation were supervised in accordance with the guidelines on supervision of dissertation laid down by the University For Development Studies.

Principal Supervisor's Signature:  Date: 24-11-2011
Name: DR. K. A. GBOLAGADE

Residue Number System (RNS) has been widely used in special purpose processors because of its interesting inherent features such as carry free addition; borrow free subtraction, digit by digit multiplication without partial product, and error detection and correction capabilities. However, RNS has not found a widespread usage in general purpose processors due to the following RNS disadvantages: sign detection, magnitude comparison, overflow detection, conversion, division etc.

In this thesis, we present an effective RNS scaler for moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$. Scaler has always been conceived as a performance bottlenecks due to the inefficient inter-modulo operation. The existing scaling algorithms have small dynamic range.

In order to accommodate application requiring larger dynamic range, we propose scaling algorithm for the three moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$. The complexity of inter-modulo operation has been dealt with by a new formulation of scaling an integer in RNS domain by one of its moduli. Chinese remainder theorem and the number theoretic properties have been exploited for this moduli set.

The proposed scheme results into an architecture that does not require any read-only memory. Another advantage of this proposal is that, the scaled integer in normal binary representation is also produced as a byproduct of this process which saves the residue-to –binary converter when the binary representation of scaled integer is also required.

Theoretically speaking, this proposal outperforms the known state of the art scalers in terms of area and delay.



I would first and foremost give praises and thanks to God for seeing me through this work.

Next, I would like to express my sincere gratitude and appreciation to my supervisor, Dr. K.A. Gbolagade for his invaluable guidance and constant encouragement throughout the progress of this thesis. He introduced me to this area of Digital Logic Design called Residue Number System (RNS), gave materials and again taught me how to research. I attribute the level of my master's degree to his encouragement and effort and without him this thesis would not have been completed or written. I say may God richly bless him.

I would like to thank Mr. Muniru M. Iddrisu who taught me coding theory course during the taught part of the MSc programme. Those theories he taught me specially Chinese Remainder Theory has really helped me in this thesis.

I would also like to thank Dr. Oyetunji, the Dean of the Faculty of Mathematical Sciences for taking us through on how to conduct a meaningful research work not forgetting my supervisor too. The knowledge acquired has helped me in carrying out this thesis.

I would like also to thank my parents, Alhaji Kassim Musah and Hamzatu Kassim, without their whole hearted support and blessings none of this would be possible. Special thanks to my wife, Rashida Salihu for her understanding, patience and enormous support throughout my Masters programme. I also thank my Mallam, Afa Issah, sister, Talahatu Kassim, brother, Abubakar S. Jibreel for their support and encouragement. May Allah bless you.

Finally, the support from my friends and colleagues is sincerely appreciated.



DEDICATION

This work is dedicated to my parents, Alhaji Kassim Musah and Hamzatu Kassim, for their support and encouragement throughout my undergraduate and graduate studies.



TABLE OF CONTENTS

DECLARATION.....	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENT.....	iv
DEDICATION.....	v
TABLE OF CONTENTS.....	vi
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
CHAPTER ONE.....	1
INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Contribution.....	4
1.3 Thesis Outline.....	5
1.4 Objectives of the study.....	5
1.5 Justification of the study.....	6
1.6 Scope of the study.....	6



CHAPTER TWO.....	7
OVERVIEW OF RESIDUE NUMBER SYSTEM.....	7
2.1 Residue Number System.....	7
2.1.1 Basic arithmetic in Residue Number System.....	7
2.1.1.1 Addition.....	7
2.1.1.2 Subtraction.....	8
2.1.1.3 Multiplication.....	8
2.2 Advantages of RNS.....	8
2.3 Disadvantages of RNS.....	9
2.4 Literature Review.....	9
2.5 Modular Adders.....	12
2.5.1 Unrestricted Moduli.....	12
2.5.2 Restricted Moduli.....	13
2.5.2.1 General module $2n+1$ addition.....	15
2.5.2.2 Carry –Save Adder.....	16
CHAPTER THREE.....	19
PROPOSED SCALING ALGORITHM.....	19
3.1 Concept.....	19



3.2	New Formulation of RNS Scaling.....	21
3.2.2	Proof of the proposed algorithm.....	22
3.2	Error Analysis.....	26
CHAPTER FOUR.....		29
HARDWARE IMPLEMENTATION.....		29
4.1	Binary representation.....	29
4.2	How to obtain y_1 , y_2 and y_3	32
4.3	Performance Analysis.....	32
4.3.1	Speed.....	32
4.3.2	Scaling errors.....	32
CHAPTER FIVE.....		35
CONCLUSION.....		34
5.1	Summary.....	34
5.2	Major contributions.....	35
5.3	Future directions.....	35
REFERENCES.....		36



LIST OF TABLES

Table 1.1: Converting integer into residue numbers.....	2
Table 2.2: Unrestricted moduli.....	13
Table 3.3: Computation traces of the proposed algorithm.....	25
Table 4.4: Algorithms and scaling errors of scalar comparison.....	33



LIST OF FIGURES

Figure 2.1: Arbitrary basic modulo-m adder.....	14
Figure 2.2: General block diagram of modulo $2n+1$ adder.....	15
Figure 2.3: Carry Save Addition process of three 5-bit operands.....	17
Figure 2.4: Carry Save Adder block.....	18



CHAPTER ONE

INTRODUCTION

1.1 Motivation

Residue Number System (RNS) is a non-weighted integer system in which addition and multiplication can be performed without interaction between residue digits in a finite ring. It is important to note that, because of this, the problem of carry propagation is eliminated and addition and multiplication can be conducted simultaneously and in parallel without interaction between digits.

RNS is defined in terms of a set of relatively prime modulo. If P denotes the moduli set, then

$$P = \{m_1, m_2, \dots, m_N\}$$

$\text{GCD}(m_i, m_j) = 1$, for $i \neq j$, where GCD represents Greatest Common Divisor.

The dynamic range M , is given by;

$$M = \prod_{i=1}^N m_i$$

Any integer in the residue class Z_M has a unique N -tuple representation given by

$$X \xrightarrow{RNS} (x_1, x_2, \dots, x_N)$$

Where $x_i = X \bmod m_i$ and is called the i th residue.

RNS represents a large integer using a set of smaller integers, so that computation may be performed more efficiently on them.



Example

If $P = \{m_1, m_2\} = (3, 5)$ and $M = 3 \times 5 = 15$ then we have the following residue numbers in the

Table below

Table 1: Conversion of integers to residues

X	\rightarrow	X_1	X_2
0	\rightarrow	0	0
1	\rightarrow	1	1
2	\rightarrow	2	2
3	\rightarrow	0	3
4	\rightarrow	1	4

X	\rightarrow	X_1	X_2
5	\rightarrow	2	0
6	\rightarrow	0	1
7	\rightarrow	1	2
8	\rightarrow	2	3
9	\rightarrow	0	4

X	\rightarrow	X_1	X_2
10	\rightarrow	1	0
11	\rightarrow	2	1
12	\rightarrow	0	2
13	\rightarrow	1	3
14	\rightarrow	2	4

RNS has significant advantages over conventional binary number systems. This is due to RNS inherent features such as carry free operations, parallelism, modularity and fault tolerance. RNS has been widely applied to generate new portable electronic devices such as camcorders, digital cameras, 3G cell phone etc. which are capable of performing Digital Signal Processing (DSP) functions.

It has become increasingly difficult to keep up with the logical culmination of the demands for higher performance and lower power budget with the increased versatility of electronic products. RNS springs up as an alternative number system of special advantages for mapping complex DSP algorithms into real time application to produce portable electronic

devices. This due to the fact that, long integer addition, subtraction, multiplication operations can be decomposed into smaller carry-free modulo operation for parallel processing or long





word length numbers can be broken up into many short word length numbers that may be operated on in parallel.

RNS coding suffers from a number of serious drawbacks all stemming from its inability to perform magnitude comparison on pairs of numbers, converting a number from RNS to binary is difficult, overflows are not easily detectable, scaling an RNS number by a constant is time-consuming, and general division is also very difficult.

Scaling is very useful for designing and implementing Inner Product Step Processor (IPSP) and iterative digital processing systems dominated by a large number of additions and multiplications. This is because, it ensures the computed results of the preceding stages do not exceed the dynamic range of the system. Being inefficient in inter-modular operations, implementing scaling operation in RNS domain entails considerable long delay and high hardware complexity. In the absence of an efficient RNS scaler, a binary scaler may be used in a hybrid RNS but it must be preceded by an expensive reverse converter. Thus it is imperative to have a high speed scaler that operates directly in RNS domain with lower complexity than the RNS-to-binary converter.

This thesis presents new adder-based RNS scaler for the moduli set $\{2^{2n} + 1, 2^n, 2^{2n} - 1\}$. In fact, the overall performance of any RNS based processor is mostly determined by the selected moduli set and the way the modulo operations i.e. addition and multiplications are implemented in hardware. The two issues are very much related as the moduli nature has a large influence on the structure and complexity of the functional units hardware. It is well

known that some moduli set example $\{2^n + 1, 2^n, 2^n - 1\}$, results in simple modular adders, it is also justified by the relative simplicity and efficiency in implementing modulo addition, multiplication and shift circuits proven by the available research publications. The conversion from the normal binary representation to residue of these moduli set is straight forward etc.



This adder-based RNS Scaler algorithm to be developed as well as its design is to compute the following: $Y = \frac{x}{k}$, where k is the scaling constant and Y is the result of scaling an integer X by k . This scaling is to be transformed into modular operations such as addition and multiplication since RNS based processor performed very well with these modular operations.

Specially, we address the following open issue in this thesis:

For applications that require larger dynamic range, can we propose an effective RNS scaler for the moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$?. What will be the implication of this in terms of area and delay?

1.2 Contribution

In this thesis, we propose solutions to the previously stated open questions. The thesis can be summarized as follows:

- For applications requiring larger dynamic range, we propose an effective RNS scaler for the moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$.
- We present a purely adder-based RNS scaler architecture.
- We demonstrate that the proposed scaler is exact and does not introduce any scaling error.

This remainder of the thesis is organized as follows;

Chapter 2 contains a brief overview of RNS. Some RNS arithmetic operations are discussed. The advantages and disadvantages of RNS are stated and the detailed literature review on scaling algorithms is also discussed. This chapter is concluded with a discussion of Modular adders focusing on implementations for: (1) arbitrary moduli and (2) restricted moduli.

Chapter 3 contains the proposed Scaling Algorithm. Formulation of New RNS Scaler is discussed; numerical example is also presented. This chapter is also concluded with a discussion of Error analysis of the proposed algorithm.

Hardware implementation is discussed in Chapter 4. The binary forms of the proposed algorithm is illustrated. The chapter is concluded with illustration of architecture of the proposed scaler.

Chapter 5 then summarizes the thesis. Major contributions and future directions are presented.

1.4 The Objectives of the study

The following are the objectives of this study

- To present an efficient scaling algorithm for moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$.
- To present scaling algorithm with minimum or no error
- To present a memoryless RNS scaler.





1.5 Justification of the study www.udsspace.uds.edu.gh

Almost all the research publications in scaling algorithm utilizes one particular moduli set which is $\{2^n - 1, 2^n, 2^n + 1\}$. Though this moduli set is very efficient, it can not be used for application that requires larger dynamic range. It is against this background that, this thesis seeks to proposed an efficient scaling algorithm with a moduli set that can be used for application that requires larger dynamic range.

1.6 The scope of the study

The scope of this thesis is restricted to scaling algorithm using the moduli set $\{2^{2n} + 1, 2^n, 2^{2n} - 1\}$. This is to be implemented in iterative digital signal processing system where scaling is very important. The scaling ensures that, the computed results of the preceding stages do not exceed the dynamic range of the system.

OVERVIEW OF RESIDUE NUMBER SYSTEM

This chapter provides fundamental information about the notions and several important properties of RNS which will give foundation to the succeeding chapters.

2.1 Residue Number System

2.1.1 RNS arithmetic operation

The standard arithmetic operation of addition/subtraction and multiplication are easily implemented with residue notation, depending on the choice of the moduli, but division is much difficult. The latter is not surprising in light of the statement above on the difficulties of sign-determination and magnitude-comparison.

2.1.1.1 Addition

This is carried out by individually adding corresponding digits, relative to the modulus. That is, a carry –out from one digit position is not propagated into the next digit position.

If the given moduli are m_1, m_2, \dots, m_N , $X \equiv (x_1, x_2, \dots, x_N)$ and $Y \equiv (y_1, y_2, \dots, y_N)$

ie. $x_i = |X|_{m_i}$ and $y_i = |Y|_{m_i}$, then we may define the addition $X + Y = Z$ by

$$\begin{aligned} X + Y &\equiv (x_1, x_2, \dots, x_N) + (y_1, y_2, \dots, y_N) \\ &= (z_1, z_2, \dots, z_N) \quad , \text{ where } z_i = |x_i + y_i|_{m_i} \\ &\equiv Z \end{aligned}$$





As an example, with the moduli-set ~~(2,3,5,7)~~ www.udspace.udsa.edu.gh, the representation of seventeen is $\{1,2,2,3\}$, that of nineteen is $\{1,1,4,5\}$, and adding the two residue numbers yields $\{0,0,11\}$, which is the representation for thirty-six in that system.

2.1.1.2 Subtraction

Subtraction may be carried out by negating the subtrahend and adding to the minuend. This is straightforward for numbers in diminished-radix complement or radix complement notation.

2.1.1.3 Multiplication

Multiplication too can be performed simply by multiplying corresponding residue digit-pairs, relative to the modulus for their position, that is, multiply digits and ignore or adjust an appropriate part of the result. If the given moduli are m_1, m_2, \dots, m_N , $X \equiv (x_1, x_2, \dots, x_N)$ and $Y \equiv (y_1, y_2, \dots, y_N)$, i.e. $x_i = |X|_{m_i}$ and $y_i = |Y|_{m_i}$ then we may define multiplication $X \cdot Y = Z$ by

$$X \cdot Y \equiv (x_1, x_2, \dots, x_N) \cdot (y_1, y_2, \dots, y_N) \\ = (z_1, z_2, \dots, z_N) \equiv Z, \quad \text{where } z_i = |x_i \cdot y_i|_{m_i}$$

2.2 Advantages of RNS

RNS has the following advantages which makes them popular when it comes to hardware implementation. For instance in digital filters, digital cameras etc.

- Long addition, subtraction and multiplication operations can be decomposed into smaller carry-free modulo operations for parallel processing.
- It can tolerate a larger reduction in supply voltage than corresponding weighted binary number system architecture for a given delay specification.

- It representation provides www.udspace.udsa.edu.gh fault tolerance as it isolates the individual digits, thus potential errors can not affect other digits
- Modularity

2.3 Disadvantages of RNS

The following are some of the disadvantages of RNS compared with their counterparts in the conventional binary system.

- Division
- Magnitude comparison
- Sign detection
- Scaling

It is important that these problems of RNS mentioned above are solved so that it can replace the binary number system fully in all hardware implementation. This thesis seeks to address one of these problems which is scaling for larger dynamic range.

2.4 Literature Review

Several techniques have been proposed on RNS scaling problem. Some of them are summarized as follows,

IShenoy and Kumaresan (1989) proposed a scaling technique based on the Chinese Remainder theorem (CRT). In their work, a redundant residue digit is introduced to enable the parallel computation of the scaled residues and a novel decomposition of CRT is applied to reduce the maximum scaling error to unit.



Griffin *et al.* (1988) first used www.advspace.mba.edu.gh CRT to produce the scaled output with error bounded by the number of modulus L used in the RNS.

Another scaling algorithm that utilizes CRT was put forward by Ulman *et al.* (1993). The later scaling method could be considered as an improved version of the design in 1967 as it eliminates the use of reluctant modulus without imposing any restrictions on the system moduli and the scaling factor.

The aforementioned scaling algorithm use CRT in such a way that the residue representation of an integer in RNS is partially converted to its binary representation before producing the scaled output.

In more recent works on scaling an integer X by a factor k for the moduli set $\{m_1, m_2 \dots m_N\}$ is based on the scaling operation defined in 1999. It states that if k is the scaling constant and Y is the result of scaling X by K then

$X = KY + |X|_k$, where $|X|_k$ denotes the remainder of X divided by k , thus

$$|Y|_{mi} = \left\lfloor \frac{x}{k} \right\rfloor_{mi} = \left\lfloor \frac{x - |x|_k}{k} \right\rfloor_{mi} \quad (xx)$$

Barsi and Pinntti (1995) derived a scaling algorithm by applying the base extension to (xx) without approximation. The later work has once again manipulated the CRT so that their design can perform both base extension and exact scaling without any system redundancy. Using almost similar scaling algorithm, Garcia and Bloris (1999) had proposed a different look-up scheme which leads to faster and simpler implementation of the RNS scaling operation than was designed in 1995.

The drawback is its restriction on the number of moduli in the system-significantly large amount of memory is required for the number of moduli. Similarly, these two designs also



use memory for implementation www.ndspace.udel.edu/~gh designs suffer from poor pipelinability and dramatic increases in hardware cost with the number of system moduli.

In 2008 The first and only full adder (FA) based exact scaling algorithm that operates directly on the residue digits was proposed by Dsygenis *et al.* It uses the axiom,

$$\|X - \|X\|_k\|_{mi} = \|x_i - \|X\|_k\|_{mi}\|_{mi} \text{ to compute the scaled output, } y_i = \left\| x_i - \|X\|_k\|_{mi} \cdot \|k^{-1}\|_{mi} \right\|_{mi} \text{ over each}$$

modulus independently, where $x_i = \|X\|_{mi}$ is a residue of the RNS defined by the moduli

set $\{m_1, m_2, \dots, m_N\}$. The computation is divided into several major stages. Each stage can be

implemented with only FAs which make the hardware cost remarkably lower and throughput

rate significantly higher than all the previous designs. Unfortunately, this algorithm has

ignored the fact that $\|X\|_k\|_{mi}$ is an inter-modular operation and it cannot be simply resolved by

treating X as an overflowed digit or residue over each modulus independently. The authors of

2008 above also made a catastrophic assumption that the scaling constant, k can be chosen to

be product of various moduli i.e., $k = \sum_{i=1}^s m_i$

With $s < N$, but it has been proven that multiplicative inverse of k for m_i ($1 \leq i \leq s$) does

not exist Barsi *et al.* (1995) and Garcia *et al.* (1999). Consequently, the output, (y_1, y_2, \dots, y_n)

calculated is not a valid RNS representation of scaling X by k .

In view of the above, the only valid adder-based RNS scaler design technique that utilizes

(xx) was proposed by Ye *et al.* (2008) and Ma *et al.* (2010). The scaler of Ye *et al.* (2008)

is dedicated to the three moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ with scaling constant 2^n while Ma *et al.*

(2010) has extended it to general moduli sets with the proviso that the moduli are co-prime

with the scaling constant. The generalization has restricted its hardware simplification,

leading to increased hardware cost and reduced performance over those of Ye *et al.* (2008)



In the late 2010, adder based RNS scaler was proposed by Shup-Hong Chang. It uses CRT and produces fast, exact RNS scaler with moduli set $\{2^n-1, 2^n, 2^n+1\}$. However this moduli set is being used by almost all the RNS scaler proposers. It is seen to be dealing with small dynamic range and it is time to look at applications that require larger dynamic range which this thesis seeks to achieve.

2.5 Modular Adders

In electronics, an **adder** or **summer** is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar. Adders can be constructed for many numerical representations.

Essentially speaking, modular adders are built using similar principles as the traditional binary adders. All improvement techniques found in binary addition can be utilized to construct modular adders. Further improvements can be obtained from the fact that the modulus is known at design time. Modulo adders can be categorized into two groups, depending on the type of modulus they are designed for:

- (1) arbitrary (unrestricted) moduli and (2) restricted moduli.

2.5.1 Unrestricted Moduli

In the first approach, the moduli can be primes or powers of primes or products of these. We begin by suing all the primes in sequence until the desired range is achieved.

To cover the range of a 16 bit binary system, a dynamic range of 65,536 is needed. RNS(17,13,11,7,5,3,2) gives dynamic range 510,510, 7.78times what is needed. The number of bits needed is $5+4+4+3+3+2+1=22$, and the efficiency is less than 1%. There are many

possible improvements. We could remove the 17, giving RNS (13, 11, 5, 3, 2), which has dynamic range 72,930, 1.11 times more than needed, and efficiency 25%.

It would be preferable to remove 17 if possible, since it is the only value that implies the (widest) column width of 5. RNS(13,11,7,5,3,2) has dynamic range 30,030, 0.458 times of what is needed etc.

These changes can continue until the required range is obtained. The summary of the above analysis is shown in Table 2 below,

Table 2: Table for unrestricted moduli

Moduli	Dynamic range	Excess Range	Widest Column	Efficiency(%)
(17,13,11,7,5,3,2)	510,510	7.78	5	<1
(17,13,11,5,3,2)	72,930	1.11	5	25%
(13,11,,5,3,2)	90,090	1.37	4	<1%
(13,11,,7,9,8)	72,072	1.1	4	25%
(13,11,63,8)	72,072	1.1	6	50%

From the table above, RNS (13,11,63,8) which has the same dynamic range as the previous version, 1.1 times what is needed, with the efficiency is 50% could be considered. However, the widest column is now 6 bits.

More unrestricted moduli sets should be exploited to have a balanced moduli set in RNS. This will also help to give RNS another look.

2.5.2 Restricted moduli

Moduli in the form 2^n-1 , 2^n , and 2^n+1 are special moduli because they greatly simplify the modular adder hardware implementation.

Designing adders for modulus 2^n requires no extra work since the correct result is always produced for both cases in Equation (1) below, by the adder in Fig1. This holds true due to the fact that the subtraction of 2^n is equivalent with dropping the carry out of the $A+B$ addition. Thus by just ignoring this carry the first adder always produces the correct results

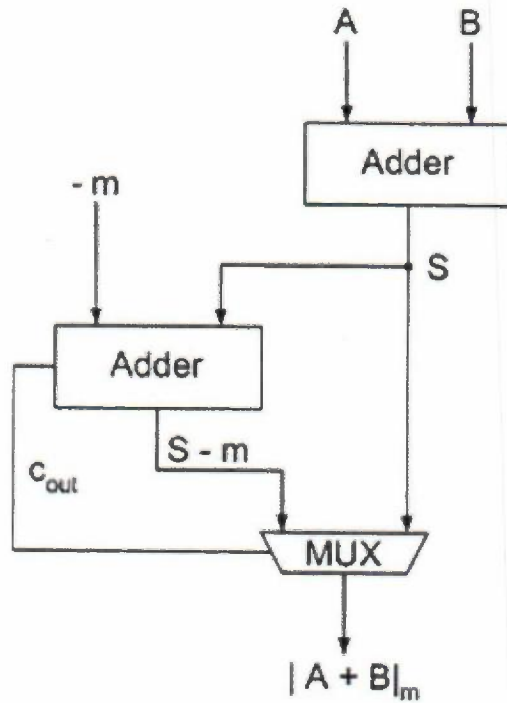


Figure 1 Arbitrary basic modulo-m adder

$$|A+B|_m = \begin{cases} A+B & \text{if } A+B < m \\ A+B-km & \text{otherwise} \end{cases}, \text{ where } k \text{ is a constant} \quad (1)$$

To design an adder with modulo 2^n-1 adder, the scheme of unrestricted adders can be used by replacing the complement m with constant 1. Essentially speaking the idea is to ignore the carry out of the $A+B$ addition but in this case an extra 1 have to be compensated somehow as by dropping the carry out we subtracted 2^n instead of 2^n-1 . By assuming that the underlying adder is a carry ripple adder and it is reused as suggested at the end of the previous section,



the correct addition is obtained by ~~reading back the carry~~ www.addspace.udsu.edu.gh generated by the addition of A and B.

2.5.2.1 General Module $2n+1$ Addition

A general block diagram, for modulo $2n+1$ adders, is depicted in Figure 2. The main function is done by the middle block, namely an abstract n -bit-long adder, where an n -bit-long operation is formally defined, for the purpose of reference as:

Definition 1 (n -bit-long operation): An n -bit unary or binary operation, performing on one or two n -bit operands and producing the result with an inconstant latency with respect to n , but at most linearly depending on n , is called an n -bit long operation. Also we define a one-step modulo $2n+1$ adder as:

Definition 2 (One-step modulo $2n+1$ adder): Any modulo $2n+1$ adder, whose critical delay path passes through only one n -bit-long binary operation is called a one step modulo $2n+1$ adder.

If the top and bottom blocks of Figure 2 perform in constant time the adder is, by a one-step adder.

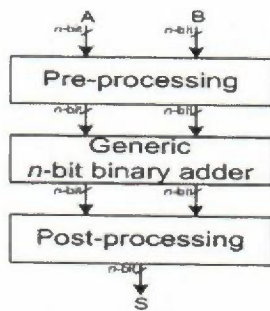


Figure 2. General block-diagram of modulo $2n+1$ adder

Examples of n -bit-long operations include:

- n -bit ripple-carry adders
- n -bit carry-accelerate adders (e.g., carry look-ahead including parallel prefix , carry-select, or carry-skip adders)



To add three numbers by hand, ~~we typically align the three~~ www.udsspace.udsu.edu.gh operands, and then proceed column by column in the same fashion that we perform addition with two numbers. The three digits in a row are added, and any overflow goes into the next column. In this case, when there is some non-zero carry, we are really adding four digits (the digits of x , y and z , plus the carry). The carry save approach breaks this process down into two steps.

First is to compute the sum ignoring any carries and separately, we can compute the carry on a column by column basis as shown in the top-row of Figure 3 below. Now, the same 'Sum' can be computed by adding the 'S' and 'C' in the final stage of the addition.

The important point is that c and s can be computed independently, and furthermore, each c_i (and s_i) can be computed independently from all of the other c 's (and s 's). This achieves our original goal of converting three numbers that we wish to add into two numbers that add up to the same sum, and in $O(1)$ time.

X:	1 0 0 1 1	X:	1 0 0 1 1
Y:	+ 1 1 0 0 1	Y:	+ 1 1 0 0 1
Z:	+ 0 1 0 1 1	Z:	+ 0 1 0 1 1
S:	0 0 0 0 1	C:	1 1 0 1 1

X:	1 0 0 1 1
Y:	+ 1 1 0 0 1
Z:	+ 0 1 0 1 1
S:	0 0 0 0 1
C:	1 1 0 1 1
Sum:	1 1 0 1 1 1

Figure 3: Carry-Save Addition Process of three 5-bit operands

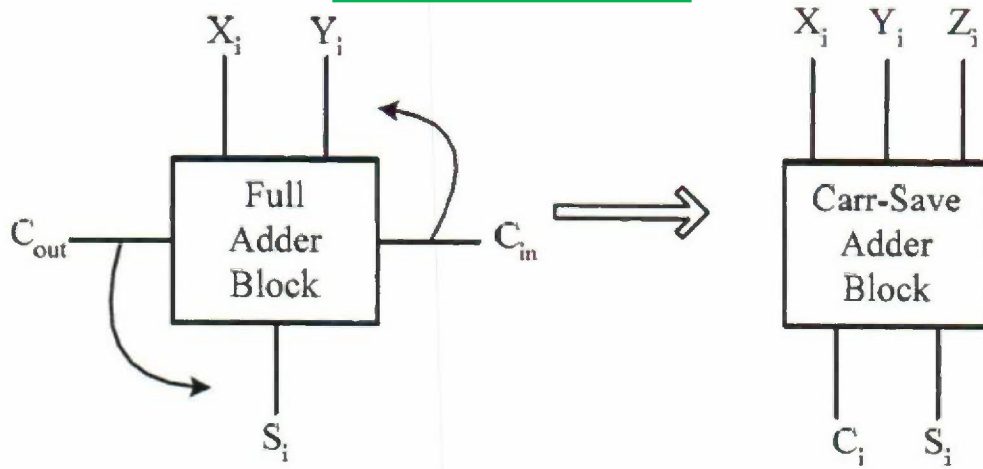


Figure 4: The carry save adder block is the same circuit as the full adde



PROPOSED SCALING ALGORITHM

3.1 Concept

Our proposed scaling algorithm is designed for the three moduli set $\{2^{2n+1}, 2^n, 2^{2n-1}\}$ in order to take care of applications requiring larger dynamic range.

Let consider an integer X which is represented by an N -tuple (x_1, x_2, \dots, x_N) with respect to a set of pair wise relatively prime numbers $\{m_1, m_2, \dots, m_N\}$, where $x_i = |X|_{m_i}$, $i = 1, 2, \dots, N$ and $|X|_{m_i}$ is defined $X \bmod m_i$

The dynamic range of the selected moduli set $\{m_1, m_2, \dots, m_N\}$ is given by

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

Based on Chinese Remainder Theorem (CRT), X is related to its residue digits by

$$X = \sum_{i=1}^N M_i |M_i^{-1}|_{m_i} x_i \bmod M \quad (2)$$

Where $M_i = \frac{M}{m_i}$ and $|M_i^{-1}|_{m_i}$ the multiplicative inverse of $|M_i|_{m_i}$

From (2), if $N = 3$, then

$$X = |m_2 m_3|_{M_1}^{-1} |m_1| x_1 + |m_1 m_3|_{M_2}^{-1} |m_2| x_2 + |m_1 m_2|_{M_3}^{-1} |m_3| x_3 \bmod M \quad (3)$$

The following axioms and theorems of addition and multiplication are used for the derivation of our scaling algorithm (Chip Hong Chang, 2010)

$$\text{Axiom1: } |X \pm Y|_m = | |X|_m \pm |Y|_m |_m$$





Axiom2: $|x \cdot y|_m = |x|_m \cdot |y|_m$ www.udsspace.uds.edu.gh

Axiom3: $A|_X|_B = A|_X|_{AB}$

Theorem1: Given the moduli set $\{m_1, m_2, \dots, m_N\}$ with $m_1 = 2^{2^n} + 1$, $m_2 = 2^n$, $m_3 = 2^{2^n} - 1$ the following holds:

$$|M_1^{-1}| = 2^{n-1} \quad (4)$$

$$|M_2^{-1}| = 2^{2^n} - 1 \quad (5)$$

$$|M_3^{-1}| = 2^{n-1} \quad (6)$$

Proof

For Equation (4)

$$|m_2 * m_3 * m_1^{-1}|_{2^{2^n}+1} = 1$$

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

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

$$\left| \left(\frac{-1}{2} \right) (-2) \right|_{2^{2n}+1}$$

$$\left| \left(\frac{-1}{2} \right) (-2) \right|_{2^{2n}+1}$$

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



Equation (5)

www.udsspace.udsu.edu.gh

$$|m_2^{-1} * m_1 * m_3|_{2^n} = 1$$

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

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

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

Equation (6)

$$|m_2 * m_1 * m_3^{-1}|_{2^{2n}-1} = 1$$

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

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

$$\left| (2) \left(\frac{1}{2} \right) \right|_{2^{2n}-1}$$

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

Theorem2: Given $p = kq$, where k is an integer,

$$| |X|_p |_q = |X|_q \quad (7)$$

3.2 New formulation of RNS scaling

Considering the above preliminaries, the following theorem is proposed to enhance the inter-modular operation of RNS scaling.

Theorem 2: let $X_i = |X|_{m_i}$, for $i = 1, 2, 3$ and $m_1 = 2^{2n} + 1, m_2 = 2^n, m_3 = 2^{2n} - 1$

If $k = m_2 = 2^n$, then

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_1} = \left\lfloor 2^n (x_2 - x_1) \right\rfloor_{m_1} \quad (8)$$

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_2} = \left\lfloor (2^{3n-1} - 2^{n-1})x_1 + (-2^{3n})x_2 + (2^{3n-1} + 2^{n-1})x_3 \right\rfloor_{m_1 m_3} \Big|_{m_2} \quad (9)$$

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_3} = \left\lfloor 2^n (x_3 - x_2) \right\rfloor_{m_3} \quad (10)$$

Proof

The formulation and the proof of the above theorem is given as follows.

By definition, scaling an integer variable X by a constant integer k can be obtained by dividing both sides of (3) by k and then taking its floor value. Let Y be the integer result of the scaling operation, using Axiom 1, we have

$$\begin{aligned} Y &= \left\lfloor \frac{X}{K} \right\rfloor \\ &= \left\lfloor \frac{1}{k} \left\lfloor m_2 m_3 \right\rfloor_{M_1^{-1}} \left\lfloor m_1 x_1 \right\rfloor_{m_1} + \left\lfloor m_1 m_3 \right\rfloor_{M_2^{-1}} \left\lfloor m_2 x_2 \right\rfloor_{m_2} + \left\lfloor m_1 m_2 \right\rfloor_{M_3^{-1}} \left\lfloor m_3 x_3 \right\rfloor_{m_3} \right\rfloor_M \\ &= \left\lfloor \left\lfloor \frac{m_2 m_3}{k} \right\rfloor_{M_1^{-1}} \left\lfloor m_1 x_1 \right\rfloor_{m_1} + \left\lfloor \frac{m_1 m_3}{k} \right\rfloor_{M_2^{-1}} \left\lfloor m_2 x_2 \right\rfloor_{m_2} + \left\lfloor \frac{m_1 m_2}{k} \right\rfloor_{M_3^{-1}} \left\lfloor m_3 x_3 \right\rfloor_{m_3} \right\rfloor_{\frac{M}{K}} \end{aligned} \quad (11)$$

where $\lfloor \cdot \rfloor$ is the least integer function

k is chosen to be $k = m_2 = 2^n$

$$\left\lfloor \frac{X}{K} \right\rfloor = \left\lfloor \left\lfloor m_3 \right\rfloor_{M_1^{-1}} \left\lfloor m_1 x_1 \right\rfloor_{m_1} + \left\lfloor \frac{m_1 m_3}{m_2} \right\rfloor_{M_2^{-1}} \left\lfloor m_2 x_2 \right\rfloor_{m_2} + \left\lfloor m_1 \right\rfloor_{M_3^{-1}} \left\lfloor m_3 x_3 \right\rfloor_{m_1 m_3} \right\rfloor \quad (12)$$





From the above Equation (12), www.udspace.udsu.edu.gh the scaled integer can be computed directly from the RNS representation of X.

As part of the objectives of this thesis to reduce or eliminate the use of converters, it is our desire to have the output of the RNS scaler to be generated directly in the residue form so that it can be directly processed by the arithmetic unit in each residue channel. The residue digit in each channel can be computed by taking the corresponding modulo operation m_i on both sides of (12) as illustrated below:

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_i} = \left| m_3 |M_1^{-1}| m_1 x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}| m_2 x_2 + m_1 |M_3^{-1}| m_3 x_3 \right|_{m_1 m_3} \Big|_{m_i} \quad (13)$$

Where $i = 1, 2$ and 3

For m_1 and m_3 channels, according to Theorem 2, (13) can be simplified to

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_i} = \left| m_3 |M_1^{-1}| m_1 x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}| m_2 x_2 + m_1 |M_3^{-1}| m_3 x_3 \right|_{m_i} \quad (14)$$

Where $i = 1, 3$

Each independently scaled residue of (14) can be further reduced to

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_1} = \left| m_3 |M_1^{-1}| m_1 x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}| m_2 x_2 \right|_{m_1} \quad (15)$$

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_2} = \left| m_3 |M_1^{-1}| m_1 x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}| m_2 x_2 + m_1 |M_3^{-1}| m_3 x_3 \right|_{m_1 m_3} \Big|_{m_2} \quad (16)$$

$$\left\lfloor \frac{X}{K} \right\rfloor_{m_3} = \left| \frac{m_1 m_3}{m_2} |M_2^{-1}| m_2 x_2 + m_1 |M_3^{-1}| m_3 x_3 \right|_{m_3} \quad (17)$$

From the above Equations, (15), www.uodsspace.uodss.edu.gh given as

$$\left| \frac{m_1 m_3}{m_2} \right| M_2^{-1} |m_2 \text{ and from}$$

$$m_1 = 2^{2n} + 1, m_2 = 2^n, m_3 = 2^{2n} - 1, \left| M_1^{-1} \right| = 2^{n-1}, \left| M_2^{-1} \right| = 2^{2n} - 1, \left| M_3^{-1} \right| = 2^{n-1}$$

We have the following,

$$\left[\left| \frac{m_1 m_3}{m_2} \right| M_2^{-1} |m_2 \right] = \left[\frac{(2^{2n} + 1)(2^{2n} - 1)(2^{2n} - 1)}{2^n} \right] \quad (18)$$

$$= 2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n$$

3.2.1 Proof of theorem

Substituting Equations (18) into (15),(16),(17), we have

First algorithm

$$\left\| \frac{X}{K} \right\|_{m_1} = \left| ((2^{2n} - 1)(2^{n-1})x_1 + (2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n)x_2 \right|_{m_1} \quad (19)$$

$$\left\| \frac{X}{K} \right\|_{m_1} = 2^n (x_2 - x_1) |_{m_1}$$

Second algorithm

$$\left\| \frac{X}{K} \right\|_{m_2} = \left| (2^{2n} - 1)(2^{n-1})x_1 + (2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n)x_2 + (2^{2n} + 1)(2^{n-1})x_1 \right|_{m_1 m_3 | m_2} \quad (20)$$



$$\left\| \frac{X}{K} \right\|_{m_2} = \left| (2^{2n} - 1)(2^{n-1})x_1 + (2^n - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n)x_2 + (2^{2n} + 1)(2^{n-1})x_3 \right|_{m_1 m_3} \Big|_{m_2}$$

$$\left\| \frac{X}{K} \right\|_{m_2} = \left| (2^{3n-1} - 2^{n-1})x_1 + (-2^{3n})x_2 + (2^{3n-1} + 2^{n-1})x_3 \right|_{m_1 m_3} \Big|_{m_2}$$

Third algorithm

$$\left\| \frac{X}{K} \right\|_{m_3} = \left| (2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n)x_2 + (2^{2n} + 1)(2^{n-1})x_3 \right|_{m_3} \quad (21)$$

$$\left\| \frac{X}{K} \right\|_{m_3} = \left| (2^n - 2^{n+1} - 2^{n+1} + 2^n + 2^n)x_2 + (2^{n-1} + (2^{n-1}))x_3 \right|_{m_3}$$

$$\left\| \frac{X}{K} \right\|_{m_3} = \left| (2^n - 2^{n+1})x_2 + (2^n)x_3 \right|_{m_3}$$

$$\left\| \frac{X}{K} \right\|_{m_3} = \left| 2^n(x_3 - x_2) \right|_{m_3}$$

ILLUSTRATION OF THE PROPOSED ALGORITHM

Example, consider the integer $X = 3316100$ with the residue representation of $(39250, 132, 39350)$ for the moduli set $\{65537, 256, 65535\}$. Using the above developed algorithm, the scaled output in RNS representation is calculated in the table3 below,

TABLE 3: COMPUTATION www.research-ujds.edu.gh TRACES OF SCALING (39250,132,39350) BY K=256 IN RNS {65537,256,65535}

$\left\lfloor \frac{X}{K} \right\rfloor_{m_1}$	$ 256(132 - 39250) _{65537} = -10014208 _{65537} = 12953 _{65537} = y_1$
$\left\lfloor \frac{X}{K} \right\rfloor_{m_2}$	$ (8388608 - 128)39250 - (2214592512) + (8388608 + 128)39350 _{4294967295} _{256}$ $= (4294980248 _{4294967295} _{256} = (12953 _{256} = 153 = y_2$
$\left\lfloor \frac{X}{K} \right\rfloor_{m_3}$	$ 256(39350 - 132) _{65535} = 10039808 _{65535} = 12953 _{65535} = y_3$

It can be verified that the RNS number (12953, 153, and 12953) is equivalent to the integer, 12953 computed directly from $\frac{3316100}{256}$. Note that the binary representation of $12953 = 0011001010011001_2$ is a byproduct.

3.3 Error analysis

The second term of the exact scaling equation of (12) is truncated to produce the approximation of

$$\left\lfloor \frac{X}{K} \right\rfloor = \left\lfloor (2^{2n} - 1)(2^{n-1})x_1 + (2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} - 2^n)x_2 + (2^{2n} + 1)(2^{n-1})x_3 \right\rfloor_{m_1 m_3} \quad (22)$$

let $k_1 = (2^{2n} - 1)(2^{n-1})$, $k_2 = 2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} - 2^n$, $k_3 = (2^{2n} + 1)(2^{n-1})$, $p = m_1 m_3$,

(22) can then be written as

$$\left\lfloor \frac{X}{K} \right\rfloor = \left\lfloor k_1 x_1 + k_2 x_2 + k_3 x_3 \right\rfloor_p \quad (23)$$

Consider the following theorem, www.udsspace.udsu.edu.gh

Theorem3: Given that $p=kq$, where k is an integer. Then

$$\left| \left| X \right|_p \right|_q = \left| X \right|_q \quad (24)$$

$$\text{but } \left| X \right|_p = X - \varepsilon p, \text{ where } \varepsilon = 0, 1, 2, \dots \quad (25)$$

Using (25), (23) can be expressed as follows:

$$\left[\frac{X}{K} \right]' = \left[\left| \kappa_1 x_1 + k_2 x_2 + k_3 x_3 - \varepsilon p \right| \right] \quad (26)$$

Without truncation, the exact Equation (15) is given by

$$\left[\frac{X}{K} \right] = \left[\left| \left(2^{2n} - 1 \right) (2^{n-1}) x_1 + \left(2^{5n} - 2^{3n+1} - 2^{n+1} + 2^{3n} + 2^n + \frac{1}{2^n} \right) x_2 + (2^{2n} + 1) (2^{n-1}) x_3 \right|_{m_1 m_3} \right]$$

$$\left[\frac{X}{K} \right] = \left[\left| \kappa_1 x_1 + \left(k_2 + \frac{1}{2^n} \right) x_2 + k_3 x_3 \right|_p \right]$$

$$\left[\frac{X}{K} \right] = \left[\left| \kappa_1 x_1 + k_2 x_2 + k_3 x_3 + \frac{x_2}{2^n} \right|_p \right] \quad (27)$$

$$\left[\frac{X}{K} \right] = \left[\left| \kappa_1 x_1 + k_2 x_2 + k_3 x_3 + c \right|_p \right]$$

$$\left[\frac{X}{K} \right] = \left[\left| \kappa_1 x_1 + k_2 x_2 + k_3 x_3 + c - \varepsilon p \right| \right]$$

$$c = \frac{x_2}{2^n}$$



Since $\left\lfloor \kappa_1 x_1 + k_2 x_2 + k_3 x_3 \right\rfloor_p = \kappa_1 x_1 + k_2 x_2 + k_3 x_3 - \varepsilon p$ is an integer, based on the definition of the least integer function, (26) and (27) can be rewritten as:

$$\left\lfloor \frac{X}{K} \right\rfloor = \kappa_1 x_1 + k_2 x_2 + k_3 x_3 - \varepsilon p \quad (28)$$

$$\left\lfloor \frac{X}{K} \right\rfloor = \kappa_1 x_1 + k_2 x_2 + k_3 x_3 - \varepsilon p + \lfloor c \rfloor \quad (29)$$

$$0 \leq x_2 < 2^n, 0 \leq c < 1, \lfloor c \rfloor = 0$$

$$\left\lfloor \frac{X}{K} \right\rfloor = \kappa_1 x_1 + k_2 x_2 + k_3 x_3 - \varepsilon p = \left\lfloor \frac{X}{K} \right\rfloor \quad (30)$$

This proves that the proposed RNS scaling algorithm is exact and does not introduce any scaling error.



HARD WARE IMPLEMENTATION

4.1 Binary representation

For Equation (8)

$$m_1 = 2^{2n} + 1, m_2 = 2^n, m_3 = 2^{2n} - 1$$

$$y_1 = \left\lfloor \frac{x}{K} \right\rfloor_{m_1} = \left\lfloor 2^n (x_2 - x_1) \right\rfloor_{m_1}$$

$$y_1 = \left\lfloor 2^n (x_2 - x_1) \right\rfloor_{2^{2n}+1}$$

$$t_1 = \left\lfloor 2^n x_2 \right\rfloor_{2^{2n}+1}$$

$$= (x_{2,n-1} x_{2,n-2} \dots x_{2,0}) \quad \text{and}$$

$$= (\overline{x_{2,n-1} x_{2,n-2} \dots x_{2,0}})$$

$$t_2 = \left\lfloor -2^n x_1 \right\rfloor_{2^{2n}+1}$$

$$= (x_{3,2n-1} x_{3,2n-2} \dots x_{3,2}, s_1, s_0) \quad \text{and}$$

$$= (\overline{x_{3,2n-2} x_{3,2n-2} \dots c_1, c_0, x_{3,2n-1}})$$

$$\text{Where, } (c_0, s_0) = \overline{x_{3,0}} + \overline{x_{3,2n}} + 1 = 0 + 1 + 1 = (1, 0)$$

$$(c_1, s_1) = \overline{x_{3,1}} + 1 + x_{3,n} = 1 + 1 + 0 = (1, 0)$$

Therefore



$$y_1 = t_1 + t_2 \quad (31)$$

The following axioms are used to develop its binary representation.

$$\text{Axiom1: } |2^{n+i} a|_{2^{n+1}} = |2^i \bar{a} + 2^{n+i}|_{2^{n+1}}$$

$$\text{Axiom2: } |2^{2^n} a|_{2^{n+1}} = |a|_{2^{n+1}}, \text{ where } a \in \{0, 1\}, i = 0, 1, n-1$$

For the Equation (9)

$$y_2 = \left\| \frac{X}{K} \right\|_{m_2}$$

$$\left\| \frac{X}{K} \right\|_{m_2} = (2^{3n-1} - 2^{n-1})x_1 + (-2^{3n})x_2 + (2^{3n-1} + 2^{n-1})x_3 \Big|_{m_1 m_3}$$

$$p_1 = |2^{3n-1} x_1|_{2^{4n-1}}$$

$$= (x_{1,n} x_{1,n-1} x_{1,n-2} \dots x_{1,0} 00 \dots 0 x_{1,2n} x_{1,2n-1} x_{1,2n-2} \dots x_{1,n+1})$$

$$p_2 = |-2^{n-1} x_1|_{2^{4n-1}}$$

$$= (11 \dots 1 \bar{x}_{1,2n} \bar{x}_{1,2n-1} \bar{x}_{1,2n-2} \dots \bar{x}_{1,0} 11 \dots 1)$$

$$p_3 = |-2^{3n} x_2|_{2^{4n-1}}$$

$$= (\bar{x}_{1,n-1} \bar{x}_{1,n-2} \dots \bar{x}_{1,0} 11 \dots 1)$$

$$p_4 = |2^{3n-1} x_3|_{2^{4n-1}}$$





$$= (x_{3,n} x_{3,n-1} x_{3,n-2} \dots x_{3,0} \overset{\text{www.udspace.uds.edu.gh}}{00\dots0} x_{3,2n-2} x_{3,2n-3} \dots x_{3,0})$$

$$p_5 = |2^{n-1} x_3|_{2^{4n}-1}$$

$$= (00\dots0 x_{3,2n-1} x_{3,2n-2} \dots x_{3,0} 00\dots0)$$

Therefore to get y_2 , then

$$y_2 = p_1 + p_2 + p_3 + p_4 + p_5 \quad (32)$$

For the Equation (10)

$$y_3 = \left\lfloor \frac{X}{K} \right\rfloor_{m_3}$$

$$y_3 = |2^n (x_3 - x_2)|_{2^{2n}-1}$$

$$y_3 = |2^n x_3 - 2^n x_2|_{2^{2n}-1}$$

$$s_1 = |2^n x_3|_{2^{2n}-1}$$

$$= (x_{3,n-1} x_{3,n-2} \dots x_{3,0} x_{3,2n-1} x_{3,2n-2} \dots x_{3,n})$$

$$s_2 = |-2^n x_2|_{2^{2n}-1}$$

$$= (\bar{x}_{2,n-1} \bar{x}_{2,n-2} \dots \bar{x}_{2,0} 11\dots1) \text{ , therefore}$$

$$y_3 = s_1 + s_2 \quad (33)$$

4.2 How to obtain y_1, y_2 and www.udsspace.udsu.edu.gh

For the y_1 , two 2n-bit Carry Save Adders (CSAs) with end-around carry (EAC) and two Modified Full Adders (MFAs) are needed to convert the inputs into diminished-one representation before the diminished $-one\ 2^{2n} + 1$ adder is used to obtain the result of y_1 .

For the y_2 , three 4n-bit CSAs with EAC, followed by $2^{4n} - 1$ adder

For the y_3 , $2^{2n} - 1$ adder can be used.

4.3 Performance analysis

The performance of the proposed scaler is evaluated by a theoretical analysis. The results of the theoretical analysis are presented as follows;

4.3.1 Speed

In fact, the speed is merely depending on the performance of the moduli $2^{2n} + 1$ and $2^{2n} - 1$ adders. In other words, the proposed scaling algorithm has successfully circumvented the complexity of RNS scaling problem for $\{2^{2n} + 1, 2^{2n}, 2^{2n} - 1\}$ by reducing it into operation as simple as addition or constant multiplication in terms of the RNS.

4.3.2 Scaling Error

For ease of reference, the algorithms and scaling errors of other designs are listed in Table... All the implementations are based on ROMs except for the proposed scaler, Dasygenis08 [8] and Garcia99 [21]. The latter is based partly on ROM-based arithmetic and partly on FAs. From the table .., it is noted that only the proposed design and the two-lookup cycle have no scaling error. The other scaling methods introduce some non-zero scaling error. Even though the error may be small, it could escalate to an unacceptable magnitude after several iterations of computation.



Table 4: Algorithms and scaling errors in scalar comparison

reference	Algorithm	Scaling error
This	FA based	0
Dasygenis 08[8]	FA based	Not applicable
Garcia 99[21]	Two lookup cycle	0
Ulman93[22]	CRT based	1
Shenoy89[25]	CRT decomposition	1.5
Griffin88[24]	L-CRT	N



CONCLUSIONS

5.1 Summary

Chapter 2 discussed fundamental information about the properties of RNS. First, Some RNS arithmetic operations are discussed. Next the advantages and disadvantages of RNS are stated and the detailed literature review on scaling algorithms is also discussed. This chapter is concluded with a discussion of Modular adders. It clear from the discussion that RNS is introduced as one of the answer to eliminate carry propagation and thus speed up the computation.

The concept of the proposed algorithm is discussed in Chapter 3. It involves the formulation of the new effective algorithm for the moduli set $\{2^{2^n}+1, 2^n, 2^{2^n}-1\}$, their proofs and concluded with examples to illustrate the application of this new scaling algorithm.

The hardware implementation of the above proposed algorithm is discussed in Chapter 4. Its binary representations as well as its architecture are also presented.

The architecture is derived from a new formulation of scaling algorithm using the Chinese Remainder Theorem and several number theoretic properties associated with this moduli set to significantly lower the complexity of inter-modular operation. The architecture can be implemented completely in full adders, making it amenable to standard cell based implementation and pipelineable to achieve very high throughput rate in a full RNS-based digital signal processor.

The integer scaled output in normal binary number representation is also generated as a byproduct of this new formulation without any additional hardware overhead. Hence



expensive residue -to-binary converter can be saved if the result after scaling is also
www.udsspace.udsu.edu.gh

required by a normal binary number system.

5.2 Major contribution

In this thesis, we present the following achievements,

- We proposed an effective RNS scaler for the moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$. For applications requiring larger dynamic range.
- We presented a purely adder-based RNS scaler architecture.
- We demonstrated that the proposed scaler is exact and does not introduce any scaling error.

5.3 Future direction

In this section, we present some future directions that could further improve RNS usage in general purpose processors.

- Proposing effective RNS scaler for the moduli sets that are free of $2n+1$ is a possible future work.
- FPGA/ASIC based implementation of the proposed scaling algorithm is left as a subject of future investigation





- Bakalist, D., and Vergos, H.T., (2009). Shifter circuits of $2n+1, 2n, 2n-1$ RNS, Electronics Letters, vol.45, no. 1, pp. 27-29.
- Barsi, F., and Pinotti, M.C., (1995). Fast base extension and precise scaling in RNS for look-up table implementation, IEEE Trans. on Signal Processing , vol.43, no. 10, pp. 2427-2430.
- Cao, B., Srikanthan, T. and Chang, C.H., (2005). Efficient reverse converters for the four-moduli sets $\{2n-1, 2n, 2n+1, 2n+1-1\}$ and $\{2n-1, 2n, 2n+1, 2n+1-1\}$, IEE Proc. Computers and Digital Tech, vol.152, no. 5, pp 687-696.
- Cardarilli, G.C. , Re, A.D., Nannarelli, A., and Re M., (2000). Reducing power dissipation in FIR Filters using the residue number system, in Pro. of the 43rd IEEE Midwest Symp. on Circuits Syst. , pp. 320-323
- Chaves, R. and Sousa, L., (2007). Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures. IET Comp. Digital Tech. , vol.5, No.1, pp.477-480.
- Conway, R. , and Nelson, J., (1999). Fast converter for 3 moduli RNS using new property of CRT,” IEEE Trans. on computers , vol. 48, no 8, pp. 852-860.
- Conway, R., and Newson, J. ,(2004). Improved RNS FIR Filter architectures, IEEE Trans. on Circuits and Syst. II , vol. 51, no.1, pp. 26-28.
- Dasygenis, M., Mitroglou, K., Soudris, D., and Thanailakis, A., (2008). A full-adder – based methodology for the design of scaling operation in Residue Number System, IEEE Trans. on Circuits and Syst. I, vol. 55, no. 2, pp. 546-558.
- Garcia, A., and Lloris, A., (1999). A look –up scheme for scaling in the RNS”, IEEE Trans. on computers, vol.48, no.7, pp.748-751.
- Gbolagade, A., (2010). Effective Reverse Conversion in Residue Number System Processors, PhD Thesis, pp. 187.
- Gbolagade, A., Cotofana, S.D., (2009). $O(n)$ Residue Number System to Mixed Radix Conversion, 2009 IEEE International Symposium on Circuits and Systems, Taipei, Taiwan, China, pp. 521-524.
- Gbolagade, A., Cotofana, S.D., (2008). A residue to Binary Converter for the $\{2n+2, 2n+1, 2n\}$ Moduli Set, 42nd Asilomar Conference on Signals, Systems, and Computers, pp. pp. 1785-1789, California, USA, October, ISBN :978-1-4244-2941-7.
- Gbolagade, A. , Chaves, R., Sousa, L.A., Cotofana, S.D., (2010). An Improved Reverse Converter for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ Moduli set, IEEE International Symposium on Circuits and Systems (ISCAS 2010), pp. 2103-2106.



Gbolagade, A., Voicu, G., www.cadence.us/du.edu/gh, Memoryless RNS-to-Binary Converters for the moduli set $\{2n+1, 2n, 2n-1\}$, 21st IEEE International Conference on Application Specific Systems Architectures, and Processors, pp. 301-304.

Gbolagade, A., Voicu, G.R., Cotofana, S.D., (2010). An Efficient FPGA Design of Reverse Converter for the Moduli Set $\{2n+1, 2n, 2n-1\}$, 5th International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems, (ACACES 2010), pp. 117-120.

Griffin, M., Sousa, M., and Taylor, F., (1989). Efficient scaling in the residue number system, in Proc. 1989 Int. Conf. on Acoustics, Speech and Signal Processing, Glasgow, Scotland, UK, pp. 1075-2078.

Griffin, M., Taylor, F., and Sousa M., (1988). New scaling algorithm for the Chinese Remainder Theorem, in Proc. Conf. Rec. Twenty-second Asilomar Conf. on Signals, Syst. and Computers, pp. 375-378.

Hiasat, A.A., (2002). High-speed and reduced-area modular adder structure for RNS, IEEE Trans. on computers, vol.51, no. 1, pp. 84-89.

Lin, S., Sheu, M., and Wang, C., (2008). Efficient VLSI design of residue to binary converter for the moduli set $\{2n, 2n+1, 2n-1\}$. IEICE Trans. INF. and SYST., vol E91-D, no. 7, pp. 2058-2060.

Mahesh, M.N., and Mehendale, M., (2000). Low power realization of residue number system based FIR filter, in Proc. of 13th Int. Conf. on VLSI Design, pp. 30-33

Meyer-base, U., Stouraitis, T., (2003.) New power-of-2 RNS scaling scheme for cell-based IC design, IEEE Trans. on VLSI Syst., vol.11, no.2, pp. 280-283.

Piestrak, S.J., (2002). Design of squarers modulo A with low-level pipelining, IEEE Trans. on Circuits and Systems-II, vol. 49, no. 1, pp. 31-41.

Re, A.D., Nannarelli, A., and Re, M., (2001). Implementation of digital filters in carry-save residue number system, in Proc. Conf. Rec. of the Asilomar Conf. on Signals, Syst. And Computers, Pacific Grove, California, USA, Nov., pp. 1309-1313

Shenoy, M.A.P., and Kumaresan, R., (1989). A fast and accurate RNS scaling technique for high speed signal processing, IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 37, no. 6, pp. 929-937.

Ulman, Z.D., and Czyzak, M., (1998). Highly parallel, fast scaling of numbers in nonredundant residue arithmetic, IEEE Trans. on Signal Processing, vol.46, no. 2, pp. 487-496.