Architecture Exploration for ALU Implementations using Multi-Modular Arithmetic

Thesis submitted by

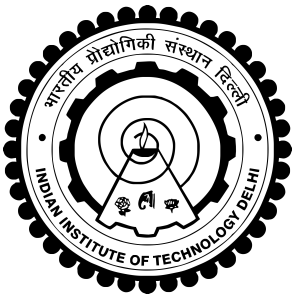
Maya Khangembam
2021EEN2020

under the guidance of

Prof. Kaushik Saha, Indian Institute of Technology Delhi

in partial fulfilment of the requirements
for the award of the degree of

Master of Technology



Department Of Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

May 2023

THESIS CERTIFICATE

This is to certify that the thesis titled **Architecture Exploration for ALU Implementations using Multi-Modular Arithmetic**, submitted by **Maya Khangembam (2021EEN2020)**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Kaushik Saha
Project Supervisor
Dept. of Electrical Engineering
IIT-Delhi, 110 016

Place: New Delhi
Date: 15th May 2023

ACKNOWLEDGEMENTS

I would like to extend my thanks to Prof. Kaushik Saha for his constant guidance and encouragement in the implementation of this project. His extensive knowledge in the field, criticisms and mentoring were critical to the completion of the project and thesis.

I also thank my classmates for their support and encouragement; my friends for making this IITD journey a joyful and memorable one; my seniors for their guidance.

Finally, the sincerest of my gratitude to my family - my father, mother, and sister for being a constant source of strength and providing unrelenting support through the entire journey.

Maya Khangembam

ABSTRACT

KEYWORDS: RNS; arithmetic circuits; cross-datapath carry propagation; residue generator

Carry propagation between datapaths for arithmetic operations in number systems based on 2's complement place a fundamental constraint on the speed, power, and area of hardware implementing it. By dividing the number into smaller parts, Residue Number System (RNS) overcomes these constraints as arithmetic operations on these parts can be performed in parallel, eliminating delay due to lengthy carry propagation chains. Systems implementing RNS, therefore, have been proposed as a means of enhancing the performance of arithmetic circuitry. However, restrictions in the expression of arithmetic operations in terms of the RNS system and overheads associated with operations involving 2's complement arithmetic make it difficult to design a programmable processor capable of benefiting from the properties of RNS. In this work we explore possible hardware efficient architectures for the conversion from binary-to-RNS and RNS-to-binary for utilising the advantages of the RNS system to achieve arithmetic operations. We also report our implementation of the converters for any general set of moduli.

Contents

List of Tables

List of Figures

ABBREVIATIONS

ASIC	Application Specific Integrated Circuit
CRT	Chinese Remainder Theorem
DSP	Digital Signal Processing
EDC	Error Detecting Code
FIR	Finite Impluse Response
GCD	Greatest Common Divisor
LUT	Lookup Table
MAC	Multiply-and-accumulate
MRC	Mixed Radix Conversion
RNS	Residue Number System
TCS	Two's Complement System

Chapter 1

INTRODUCTION

1.1 Residue Number System (RNS)

The most popular number system choice in the past several decades for design of digital units in a variety of digital systems, including microprocessors, application specific integrated circuits (ASIC), digital signal processors (DSP), mainframe computers etc, is the Two's Complement Number System (TCS) which is a binary numeral system for representation of signed integers. Consequently, numerous research works aimed towards the enhancement of performance of digital blocks implementing TCS exist. Digital designers of date must investigate all design abstraction levels, from the algorithmic level up to even the logic or transistor level, to find alternatives for increasing speed and decreasing power consumption in order to keep up with the incessant pursuit for high-speed-low-power DSPs for higher versatility in application areas. This has been the primary reason for abandoning programmability in favor of hardware implementations dedicated to execution of complex DSP algorithms, and it remains one of the most important standards in modern digital IC design. Systems implementing TCS face large datapath delay primarily due to arithmetic blocks involving carry propagation from previous datapaths for computation. This timing bottleneck stems from TCS's default positional weighting. Over the years, researchers and designers have developed a multitude of ingenious strategies to reduce the TCS-related carry propagation delay. Parallel prefix adder, sparse tree adder, carry lookahead adder, carry select adder, and carry skip/bypass adder are examples of well-known methods for accelerating the carry propagation addition of two operands. All of these high-velocity adders sacrifice power and area for velocity. In high-speed structures, both the fanouts of used primitive logic gates and the complexity of implemented hardware are typically very large, resulting in high amount of power dissipation.

An alternative is the Residue Number System (RNS), a non-weighted number system, which presents itself as a viable solution for implementing efficient hardware that computes arithmetic operations common to DSP algorithms such as convolution which consists of numerous multiplication and addition operations (1). In the RNS domain, numbers of sizeable word-length are decomposed into counterparts of smaller word-length by taking the remainder of the large number w.r.t. a selected set of moduli and the original number is thus represented using the remainders obtained. In their respective modulus channels, arithmetic operations such as addition, subtraction, and multiplication can be executed independently and in parallel on the obtained reduced word-length remainders, thus speeding up the time

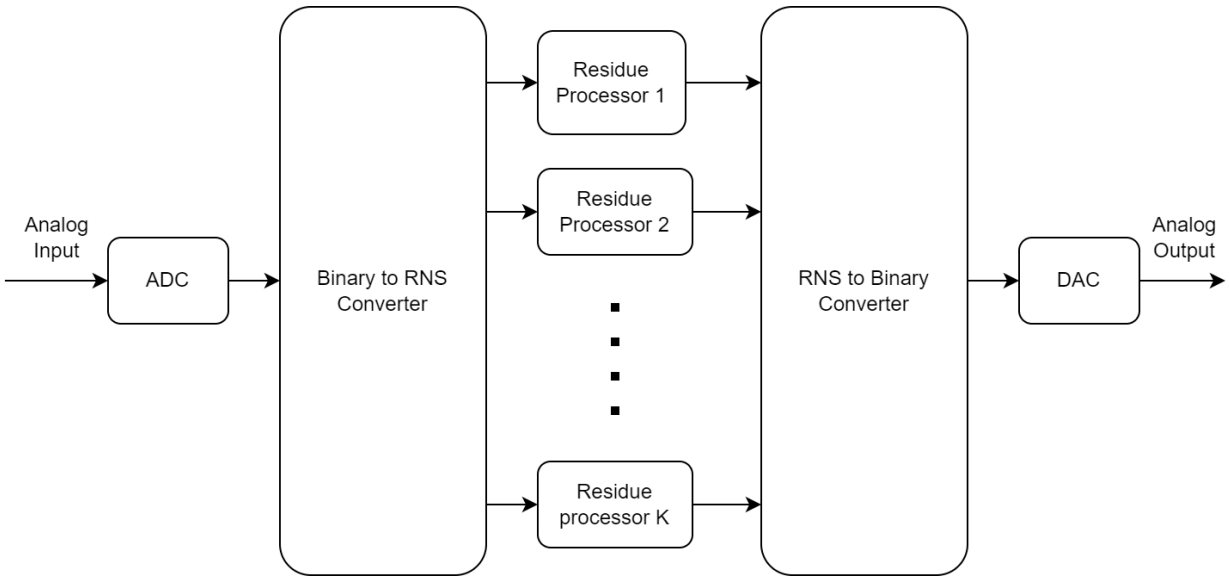


Figure 1.1: An example of a basic signal processor using RNS

taken to perform these operations. Due to the introduction of sub-word length parallelism into the algorithm design and breaking of the carry propagation chain, RNS-based digital blocks outperform their TCS counterparts in terms of performance. Relatively lower static power and dynamic power dissipation can be achieved in architectures implemented in RNS as design methodologies such as multi-threshold voltage cells and voltage scaling for dynamic frequency and voltage scaling are more conveniently utilized due to it being comparatively easier to meet a target timing specification for such architecture. Moreover, as a result of their modularity and parallelism, arithmetic units based on RNS have more area efficiency, lesser localized interconnections complexity as well as lesser switching activity, resulting in a greatly reduced total power. Thus, the typical RNS system can be implemented satisfactorily in applications where the use of comparisons, divisions and scaling operations are less and can be avoided. Such applications include FIR filter, image encoding and encryption, spread spectrum communication systems which require low power and high speed designs. Fig. 1.1 illustrates a basic signal processor using RNS. An analog-to-digital converter and binary-to-RNS converter constitute the forward conversion. The binary-to-RNS converter outputs k remainders which are then parallelly processed by k processors. The back conversion consist of the RNS-to-binary converter along with a digital-to-analog converter which convert the k residue words to the final analog output.

1.2 Motivation

Despite its benefits, RNS has limited application in the design of general-purpose digital signal processors. RNS suffers from inefficient inter-modulo operations due to the unweighted

nature of its algebraic structure. Inter-modulo operations including magnitude comparison, division, scaling, sign and overflow detection, and RNS-to-binary conversion requires the outputs from all moduli channels in order to evaluate the correct result. This is in contrast to the operations of addition, subtraction, and multiplication, which do not require this information. The elimination of these operations' modularity and parallelism characteristics results in the hardware implementation of these operations having a relatively high area requirement and a slower performance compared to systems implemented using TCS. Because of this, RNS has been well received in a number of applications, particularly ones in which multiplications and additions or subtractions dominate the datapath.

The complexity and performance of the implemented hardware is directly and substantially affected by the selection of the base moduli to be used for RNS. Special moduli, such as that of the form 2^n and $2^{n\pm1}$, have favorable number-theoretic properties that allow for the efficient implementation of RNS conversion and operations, particularly inter-modulo operations which are hardware intensive (2), (3). However, as the moduli to be used in the RNS are required to be relatively prime to each other, obtainable values of dynamic range are restricted as the RNSs constructed with moduli of 2^n and/or $2^{n\pm1}$ due to the finite size and number of moduli sets that can be constructed which fulfill the requirements. The size of one or more special moduli must be changed to change the dynamic range, which in turn degrades the overall performance of the system.

Studies have indicated that computations performed in the RNS domain offer significant tolerance to delays caused by process-induced parameter variations when the suitable moduli set is used. Larger modulo operations are found to increase circuit timing uncertainty as well as cause greater delay variation along the datapath on which they reside. From a variation-tolerant standpoint, RNS composed of numerous small-sized balanced moduli are preferred. Moreover, it is demonstrated that the cardinality of the moduli set ceases to affect the complexity of the RNS-to-binary converter past a particular threshold. Consequently, to expand the dynamic range of an RNS, increasing the cardinality is more favorable than enlarging individual moduli. Valid moduli sets for the RNS can be constructed relatively easily by selection of an ample number of moduli as desired from a pool of small integers that satisfy the requirements of dynamic range and relative primality.

The efficiency of modular arithmetic operations can adequately satisfy most needs if the arbitrary modulus size is confined to a word-length of not more than six bits, which can be easily achieved by augmenting the size of the moduli set. Even for a high-cardinality and balanced RNS, the throughput is found to decrease with increase in the number of arbitrary residues required to be generated for each input operand increases. The time required for computation of the residues for different moduli sets and varying operand size typically exhibits significant variability owing to differences in periodicity of the involved moduli(4).

1.3 Objective

The main focus of this work was to develop efficient and area optimised methods of residue generation for a given arbitrary module as well as fast conversion methods to reduce area as well as delay overhead that is caused due to the generation of moduli and conversion of the numbers between the two number systems.

The following key issues were identified, and research was conducted to achieve the primary aim of this study:

- To investigate hardware implementation issues of converters employing the Chinese Remainder Theorem and Mixed Radix Conversion algorithms as well as variations in residue generation for different moduli.

- To analyse hardware efficient and minimize disparity in architectural design between moduli of varying cyclic periodicity for computation of residue for a given arbitrary module and generation of residue values from binary value and back.

1.4 Project Plan

The project work was planned to be implemented in the following stages:

Stage 1 was focused on collection and study of materials for overall idea of existing standard methods for conversion from binary to RNS and back. Understanding the different theorems, their proofs, differences in properties and execution based on different moduli set, their advantages and disadvantages and their hardware implementation.

Stage 2 was aimed at exploring hardware efficient approaches of implementing arbitrary residue generators, alternatives to the traditional CRT and MRC methods for conversion between the two different systems.

Stage 3 was aimed at writing and compiling of code including MATLAB code for generating information and modules required for the residue generators and the forward and back conversion between RNS and binary systems, the integration of the modules to obtain a working system model as well as comparison between performances of system models implemented using different methods.

1.5 Thesis Outline

The remainder of this thesis is ordered thus:

- Chapter 2 gives an overview of the different methods for residue generation and conversion of a given decimal number from bianry to RNS and vice-versa.
- Chapter 3 provides a thorough discussion of the algorithm, implemented system model, interface and the connections of different modules.
- Chapter 4 includes the comparision of area, delay and power between the synthesized implemented model versus the traditional methods and issues in the current model.
- Chapter 5 presents the overall conclusion of the thesis and reviews the improvements to be done as well as the future scope of this project work.

Chapter 2

Background

This chapter provides an overview of the RNS system and algorithms for conversion to binary and back as used in the project work.

2.1 Modular Arithmetic

A set of integers and arithmetic operations performable on this set consist an integral numeral system. For a given base r_i called radix and a finite set of digits of size r_i , $(d_1, d_2, \ldots, d_N)$, any number P can be denoted as

$$P = \sum_{i=0}^{N-1} x_i d_i$$

where p_i is the positional weight at the i-th position corresponding to the digit d_i and is equal to the i-th power of the radix, i.e., r_i and N is the word-length of P. Thus, the number distinct digits that can be used for the representation of p_i is r_i. It can be then seen that the contribution of each digit d_i to the integer is dependent on the position of the digit and a representation of this manner for a numeral system is termed as positional. Another term for such numeral systems is the weighted number system.

If, for different digit positions, the different radices r are used then the number system is known as a mixed-radix system. Similary a numeral system is called fixed-radix if all the N digit positions of the integer P use the same radix r_i. Examples of the fixed-radix system include the decimal number system which has a radix of 10 and is the standard number system used worldwide, and the binary number system, radix of 2, and is the popular choice for electronic and computer computation hardware.

Other frequently employed fixed-radix number systems include octal (radix of 8) and hexadecimal (radix of 16). As each position carries a corresponding weight in the weighted number system, the hardware for numerous core operations can be efficiently implemented. For example, division or multiplication by varied factors of 2 is simply implementable by the use of a simple programmable shifter, in the binary number system. Additionally, implementation of sign and overflow detection in this number system is achievable by just the observation of the most significant bit of the number and the carry output bit, respectively.

However, a disadvantage of the positional number system is that the delay due to inter-digit carry propagation of the operand places a constraint on the speed of arithmetic operations as the value of every digit of higher significance is dependent on all digits of lesser or equal significance of the operands. This issue can be avoided using the Residue Number System (RNS) which is a weightless number system. In the RNS domain, the length of the carry propagation chain is reduced by dividing it into numerous smaller length chains to accelerate addition, subtraction, and multiplication computations.

To understand RNS, we must first look at the fundamentals of modular arithmetic. Given two numbers X and A, the modulo operation $x = X \bmod A$ or $x = [X]_A$ is defined as

$$X = x + mA$$

for some integer m such that $0 \leq m \leq A$ and can be represented as X_A. x or X_A is also known as the remainder of the division operation $\frac{X}{A}$ while m is the integer quotient. Two integers P and Q are in the same equivalence class or called to be congruent, modulo N provided they produce the same remainder upon division by N. The relation is then denoted as

$$P \equiv Q \pmod{N}$$

and implies $P - Q = kn$ where k is an integer.

For a set of moduli

$$(m_1, m_2, \ldots, m_i, \ldots, m_n)$$

coprime to each other, that is, $\mathrm{GCD}(m_i, m_j) = 1$ for $i \neq j$, the product all moduli m_i

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

is known as the dynamic range of the RNS. Thus, for any integer $X \in [0, M-1]$ a set of residues below exists as a unique representation in the RNS.

$$X \xrightarrow{\text{RNS}} ([X]_{m_1}, [X]_{m_2}, \ldots, [X]_{m_n})$$

where $X < M$.

The mapping of an integer in the decimal system to RNS involves consideration of two cases:

- **Case I.** Even M: the range of the RNS is given by:

$$\frac{-M}{2} \leq X \leq \frac{M}{2} - 1$$

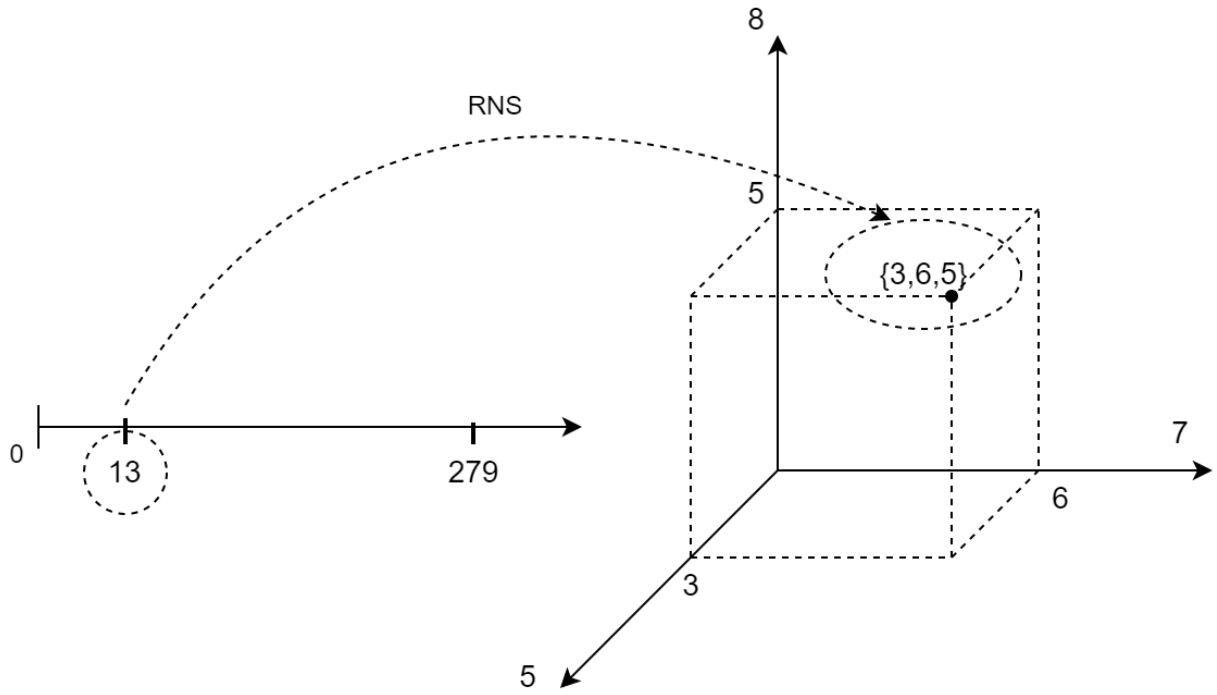


Figure 2.1: An illustration of mapping from decimal to RNS

- **Case II.** Odd M: the range of the RNS is given by:

$$-\frac{M-1}{2} \leq X \leq \frac{M-1}{2}$$

For negative numbers, the residues can be obtained by complementing the residue as obtained for the positive numbers, i.e,

$$[X]_{m_i} = \begin{cases} [X]_{m_i} & \text{for} X \geq 0 \\ [m_i - [X]_{m_i}]_{m_i} & \text{for} X < 0 \end{cases}$$

Arithmetic operations such as multiplication and addition can then be parallelly done for different moduli to obtain the required outcome:

$$Z = X \, op \, Y \xrightarrow{\text{RNS}} \begin{cases} Z_{m_1} = [X_{m_1} \, op \, Y_{m_1}]_{m_1} \\ Z_{m_2} = [X_{m_2} \, op \, Y_{m_2}]_{m_2} \\ \vdots \\ Z_{m_n} = [X_{m_n} \, op \, Y_{m_n}]_{m_n} \end{cases}$$

Mentioned below are some properties for exposition of the theorems used in this thesis:

Property 1: $[mA]_m = 0$

Property 2: $|A + B|_m = ||A|_m + |B|_m|_m$

Property 3: $|A \cdot B|_m = ||A|_m \cdot |B|_m|_m$

Property 4: If the the k least significant bits of A is denoted as $A[k-1:0]$, then

$$[A]_{2^k} = A[k-1:0]$$

By incorporating parallelism at the sub-word level, the delay of modulo arithmetic blocks is lower as compared to ones implemented in binary. Special moduli sets can be used to achieve even further high-speed arithmetic blocks leading to high reception of special moduli set based RNS's in applications involving multiple additions/subtractions and multiplications. DSP algorithms generally contain sum-of-products kernels which can be implemented using RNS-based arithmetic units. Thus, DSPs based on RNS can achieve higher performance.

2.2 Choice of Moduli

The selection of moduli in an RNS is crucial due to its effect on system processing speed, power consumption and complexity. In addition, finding a straightforward method for the selection of moduli is not easy as there are multiple constraints which may clash with each other. For instance, for the maximization of parallelism, the required dynamic range must be achieved using as many moduli possible, however such a choice conflicts with the complexity of implementation of some operations; for example, the MRC. Several crucial criteria are outlined and briefly illustrated in the following sections.

- The moduli in the selected set must be coprime.
- The dynamic range of the selected moduli selected be more than the range required by the application.
- For a balanced and comparable hardware speed and complexity of the different moduli channel in an RNS-based processor, the moduli should be selected such that any disparities in value and properties is minimal.
- Due to the fact that only one module can be a power of two and the other modules are not powers of two, the selection of moduli should correspond to the lowest possible coding overhead.
- Moduli of the powers of two or close, such as $m_i = 2^{k-1}, 2^k, 2^{k+1}$, are known as special moduli and are useful for simplifying modular addition and multiplication.

- Smaller values of individual moduli should be used to increase the speed of the RNS-based system due to the parallel characteristic of the RNS.

- Similarly, choosing a set of moduli with high cardinality enables higher-speed implementation due to parallelism.

- A moduli selection characterized by the maximum number of unity multiplicative inverses is advantageous for simplifying certain RNS operations, such as MRC and CRT.

2.3 Input/Output Conversion

This section illustrates the general methods for the input and output conversion in RNS systems. The conversion to and fro between RNS and decimal/binary system add a significant hardware as well as delay overhead to systems implemented in RNS. However this can disregarded in cases requiring multiple arithmetic operations as with increase in the number of operations performed in the RNS domain, the load due to overhead from input and output conversion decreases in the RNS processor. Several literary works exist on efficient methods to achieve these conversions.

2.3.1 Input Conversion

To convert the input data into RNS format, the integer remainder of the number should be computer when divided by m_i as mentioned previously, Several literary works exist on efficient conversion and can be classified into mainly two groups - general moduli sets and special moduli sets. While special moduli present the advantage of fast and easy computations required for conversion, the constraint of co-primality in addition to the limited number of moduli of this form result in difficulty of selection of such moduli for a given dynamic range. Thus, general moduli sets are preferred and a straight-forward way of implementing such converters is by the means of LUTs. If the input is an p-bit number, the size of the look-up table (LUT) required can be calculated as 2^p, and the number of bits required for residue-to-binary conversion is given as

$$z = \sum_{i=0}^{n} \lceil log_2 m_i \rceil$$

,where n is the size of the moduli set used and m_i is the ith moduli.

This is an efficient approach when the size of the number undergoing conversion to RNS, i.e, p, is small. Selection of an appropriate moduli set also helps in minimizing coding overhead,

thereby, reducing usage of memory resources. For large p, the size of the LUT required is substantial and infeasible. In such cases, the distributive property of residue numbers can be used so that the modular sum of the powers of two which are weighted by the bits of the input number gives the residue of the number w.r.t. the moduli set. However, such methods have relatively high amounts of delay which is unsuitable for high-speed designs.

2.3.2 Output Conversion

Even for the special moduli set, the lack of modularity and parallelism in backward conversion means that an RNS-to-binary converter typically results in a much longer delay and larger hardware area than a binary-to-RNS converter. Fortunately, however, the gain in speed achieved due to computation of a large number of MAC operations directly in the RNS domain typically significantly offsets the penalty to area and performance of the RNS-to-binary converter as, in a purely RNS system, the conversion from RNS to binary is only required once in the computation the final output and is thus a negligible proportion of the total system cost.

General methods for conversion from RNS to binary include the CRT (Chinese Remainder Theorem) and MRC (Mixed Radix Conversion). CRT, in particular, is popular for high speed implementations.

Chinese Remainder Theorem

The Chinese remainder theorem is used to find a unique solution for simultaneous linear congruences with coprime moduli, that is, for given a set of moduli $(m_1, m_2, \ldots, m_n)$ coprime to each other and some integers $x_1, x_2, \ldots, x_n$, the system of simultaneous congruences

$$\begin{aligned} X &\equiv x_1 \pmod{m_1} \\ X &\equiv x_2 \pmod{m_2} \\ &\vdots \\ X &\equiv x_n \pmod{m_n} \end{aligned}$$

has a unique solution under modulo $M = \prod_{i=1}^{n} m_i$. The solution X to this system of congruences can be found using a general algorithm as given below.

1: Compute $M = \prod_{i=1}^{n} m_i$

2: Compute

$$y_i = \frac{M}{m_i} = m_1 m_2 m_{i1} m_{i+1} m_k$$

for each $i = 1, 2, \ldots, k$

3: Compute the modulo inverses $\left|1/y_i\right|_{m_i}$ as

$$y_i \cdot \left|\frac{1}{y_i}\right|_{m_i} \equiv 1 \,(\mathrm{mod}\, m_i)$$

for each $i = 1, 2, \ldots, k$ using Euclid's extended algorithm (z_i exists since $m_1, m_2, \ldots, m_n$ are pairwise coprime).

4: Compute

$$X = \left[\sum_{i=1}^{n} x_i y_i \left|\frac{1}{y_i}\right|_{m_i}\right]_M$$

to get a unique integer solution to the system of congruences modulo N.

If the RNS system is required to operated with both positive and negative integers, the output conversion algorithm must be slightly modified in order to convert the residues to a binary range of positive and negative values. Due to the algorithm used for mapping the relative integers, as mentioned previously, the following rule must be used to determine positive and negative binary values:

- If the value of $Z = \sum_{i=1}^{n} x_i y_i \left|\frac{1}{y_i}\right|_{m_i}$ is less than or equal to the range of signed representation then $[Z]_M$ is required positive value of X.

- If the value of Z is greater the range of signed representation then $[Z]_M - M$ is required negative value of X.

Mixed Radix Conversion

Another popular method for finding a unique solution X for a given system of congruences is the mixed radix conversion, the general algorithm for which has been included below.

1: Compute modulo inverses $\left|1/y_i\right|_{m_j}$ for $1 \leq i < j \leq n$ where

$$m_i \cdot \left|\frac{1}{m_i}\right|_{m_j} = 1 \pmod{m_j}$$

2: Compute the mixed radix digits z_i as

$$z_1 = [x_1]_{m_1}$$
$$z_2 = \left[(x_2 - z_1)\left|\frac{1}{m_1}\right|_{m_2}\right]_{m_2}$$
$$z_3 = \left[\left((x_3 - z_1)\left|\frac{1}{m_1}\right|_{m_3} - z_2\right)\left|\frac{1}{m_2}\right|_{m_3}\right]_{m_3}$$

$$\vdots$$

$$z_n = \left[\left(\ldots \left((x_n - z_1) \left| \frac{1}{m_1} \right|_{m_n} - z_2 \right) \left| \frac{1}{m_2} \right|_{m_n} - \ldots - z_{n-1} \right) \left| \frac{1}{m_{n-1}} \right|_{m_n} \right]_{m_n}$$

3: Compute

$$X = z_1 + z_2 m_1 + z_3 m_1 m_2 + \ldots + z_n m_1 m_2 \ldots m_{n-1}$$

As can be observed, MRC is a sequential process. The coefficient z_i can only be computed if coefficients z_1 to z_{i-1} have been computed. Thus, the MRC approach has a delay of $O(n)$. Such delay is not desirable for high-speed arithmetic applications where the CRT is intend to be used.

Chapter 3

Methodology, Approach and Tools

3.1 System Model

In this section, an outline of the modules and sub-modules used in this project consisting of different binary-to-RNS and RNS-to-binary converter architectures as well as residue generator modules has been provided.

3.1.1 Architecture

The general architecture for the implemented binary-to-RNS-to-binary converter is as depicted in Fig. 3.1. The architecture was implemented in several different ways such as using the traditional CRT, new CRT I, period based residue generators and half-period based residue generators for comparing performances of the converters.

3.1.2 Interfaces

The overall binary-to-RNS converter has the inputs reset, clk and the number to be converted N and outputs mod_m1 to mod_mn which are the residue output values. Similarly, the overall RNS-to-binary converter has the inputs mod_m1 to mod_mn and the output N_out which is the final binary values. Modules for computation in the RNS domain can be inserted between these two converters.

The interfaces of the used modules have been encapsulated in Table 3.1, 3.2 and 3.3.

3.2 Implemented Methods

3.2.1 New Chinese Remainder Theorem I

Before mentioning the new CRT I theorem as proposed by Wang (5), the following propositions must be listed first.

Proposition 1: $[am_1]_{m_1 m_2} = [a]_{m_2} \cdot m_1$

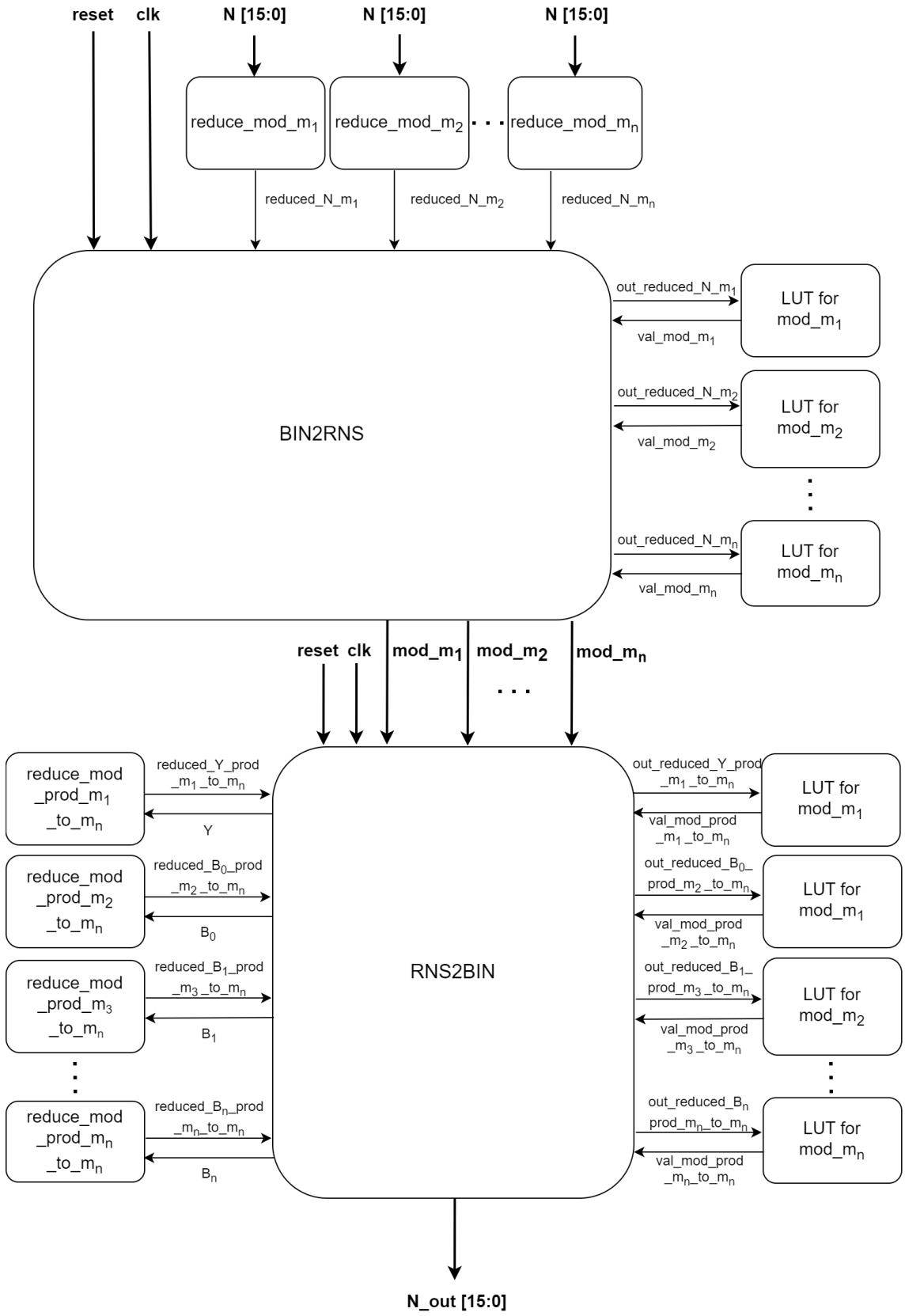


Figure 3.1: General architecture of the implemented converters B_i

reduce_mod_mi or reduce_mod_prod_mi_to_mn		
Signal Name	I/O	Description
X	input	input value for which RNS equivalent in the modulus m is to be calculated
reduced_N_mi	output	value of reduced bit-size and of same residue value as of input value X when taken mod m_i

Table 3.1: Table of ports of module reduce_mod_mi/reduce_mod_prod_mi_to_mn

BIN2RNS		
Signal Name	I/O	Description
reset	input	reset signal for initialisation and reset of module
clk	input	clock signal input
reduced_N_mi	input	value of reduced bit-size and of same residue value as of output value X when taken mod m_i
val_mod_mi	input	residue of reduced_N_mi w.r.t. m_i
out_reduced_N_mi	output	value obtained from output of reduce_mod_mi to be sent to LUT
mod_mi	output	output signal of value N modulo m_i

Table 3.2: Table of ports of module BIN2RNS

Proposition 2: $p = 1 \bmod m_1 m_2 \ldots m_k \Leftrightarrow p = 1 \bmod m_1 , p = 1 \bmod m_2, \ldots, p = 1 \bmod m_k$

Proposition 3: The mixed radix form can be used to uniquely represent any integer $a \in [0, M-1]$ as

$$a = a_0 + a_1 m_1 + a_2 m_1 m_2 + \ldots + a_{n-1} m_1 m_2 \ldots m_{n-1}$$

where a_i is the i-th coefficient in the mixed radix representation and is $0 \leq a_i \leq m_{i+1}$.

Additionally, to establish the new CRT I the following theorem must be mentioned.

Theorem: For a given residue representation of any number $X = (x_1, x_2, \ldots, x_n)$ and set

RNS2BIN		
Signal Name	I/O	Description
`reset`	input	reset signal for initialisation and reset of module
`clk`	input	clock signal input
`mod_mi`	input	residue values modulo m_i to be converted back to binary
`reduced_Z_prod_mi_to_mn`	input	bit-size reduced value of Z (= Y or first-order radix elements B_i) taken modulo $M = \prod_i^n m_i$
`val_mod_Z_prod_mi_to_mn`	input	residue of `reduced_N_mi` w.r.t. M
`N_out`	output	final binary value which is equivalent to the RNS number (mod_m1, mod_m2, ..., mod_mn) in the moduli set $(m_1, m_2, ..., m_n)$
`out_reduced_Z_prod_mi_to_mn`	output	value obtained from output of `reduce_mod_m` to be sent to LUT
`Y`	output	N_out = Y mod M
`B_i`	output	elements of first-order radix B

Table 3.3: Table of ports of module `RNS2BIN`

of moduli $(m_1, m_2, \ldots, m_n)$, X has the following relation with its RNS representation

$$X = [x_1 + k_1 m_1 (x_2 - x_1) + k_2 m_1 m_2 (x_3 - x_2) + \ldots + k_{n-1} m_1 m_2 \ldots m_{n-1} (x_n - x_{n-1})]_{m_1 m_2 \ldots m_{n-1} m_n}$$

where the modulo inverse of $\left[\prod_{j=1}^{i} m_i\right]_{m_{i+1} m_{i+2} \ldots m_n}$ is given by k_i, that is,

$$\begin{aligned} k_1 m_1 &= 1 \pmod{m_2 m_3 \ldots m_n} \\ k_2 m_1 m_2 &= 1 \pmod{m_3 m_4 \ldots m_n} \\ &\vdots \\ k_{n-1} m_1 m_2 m_3 \ldots m_{n-1} &= 1 \pmod{m_n}. \end{aligned}$$

The equation defined by the theorem is a mixed radix representation but it is different from

the MRC algorithm in the fact that unlike the sequential process of MRC, the coefficients are based on the indices as in the CRT. Thus, it has the advantages of both the CRT as well as the MRC.

For moduli set given as $(m_1, m_2, \ldots, m_n)$, the set $a_1, a_2, \ldots, a_n$ is uniquely defined and can be calculated as,

$$\begin{aligned} a_0 &= [1 - k_1 m_1]_{m_1 m_2 \ldots m_n} \\ a_1 &= [k_1 - k_2 m_2]_{m_2 m_3 \ldots m_n} \\ &\vdots \\ a_{n-2} &= [k_{n-2} - k_{n-1} m_{n-1}]_{m_{n-1} m_n} \\ a_{n-1} &= [k_{n-1}]_{m_n} \end{aligned}$$

which can then be represented in the mixed radix representation such that a_0 is represented in MRC for the moduli set $(m_1, m_2, \ldots, m_n)$, a_1 in the moduli set $(m_2, m_3, \ldots, m_n)$, and so on till $a_n n - 1$.

$$\begin{aligned} a_0 &= a_{0,0} + a_{0,1} m_1 + \ldots + a_{0,n-1} m_1 m_2 \ldots m_{n-1} \\ &\qquad\qquad 0 \leq a_{0,i} \leq m_{i+1} \\ a_1 &= a_{1,1} + a_{1,2} m_2 + \ldots + a_{0,n-1} m_2 \ldots m_{n-1} \\ &\qquad\qquad 0 \leq a_{1,i} \leq m_{i+1} \\ &\vdots \\ a_{n-2} &= a_{n-2,n-2} + a_{n-2,n-1} m_{n-1} \\ &\qquad\qquad 0 \leq a_{n-2,i} \leq m_{i+1} \\ a_{n-1} &= a_{n-1,n-1} \\ &\qquad\qquad 0 \leq a_{n-1,n-1} \leq m_{i+1} \end{aligned}$$

For the given moduli set $(m_1, m_2, \ldots, m_n)$, the characteristic matrix is then represented as

$$A = \begin{pmatrix} a_{0,0} & 0 & \ldots & 0 & 0 \\ a_{0,1} & a_{1,1} & \ldots & 0 & 0 \\ & & \ddots & & \\ a_{0,n-2} & a_{1,n-2} & \ldots & a_{n-2,n-2} & 0 \\ a_{0,n-1} & a_{1,n-1} & \ldots & a_{n-2,n-1} & a_{n-1,n-1} \end{pmatrix}$$

where $a_{i,j} < m_{i+1}$. Then for the given RNS representation $(x_1, x_2, \ldots, x_n)$ for the number

X, the vector B can be defined as the first-order radix of the RNS number X and is given by

$$B = A \cdot X'$$

$$\begin{pmatrix} B_0 \\ B_1 \\ \vdots \\ B_{n-2} \\ B_{n-1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & 0 & \dots & 0 & 0 \\ a_{0,1} & a_{1,1} & \dots & 0 & 0 \\ & & \ddots & & \\ a_{0,n-2} & a_{1,n-2} & \dots & a_{n-2,n-2} & 0 \\ a_{0,n-1} & a_{1,n-1} & \dots & a_{n-2,n-1} & a_{n-1,n-1} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}$$

The elements of matrix B, B_i is the i-th pseudodigit and equals $a_{0,i}x_1 + a_{1,i}x_2 + \ldots + a_{i,i}x_{i+1}$ for $i = 0, 1, \ldots, n-1$ as can be obtained from the matrix multiplication above.

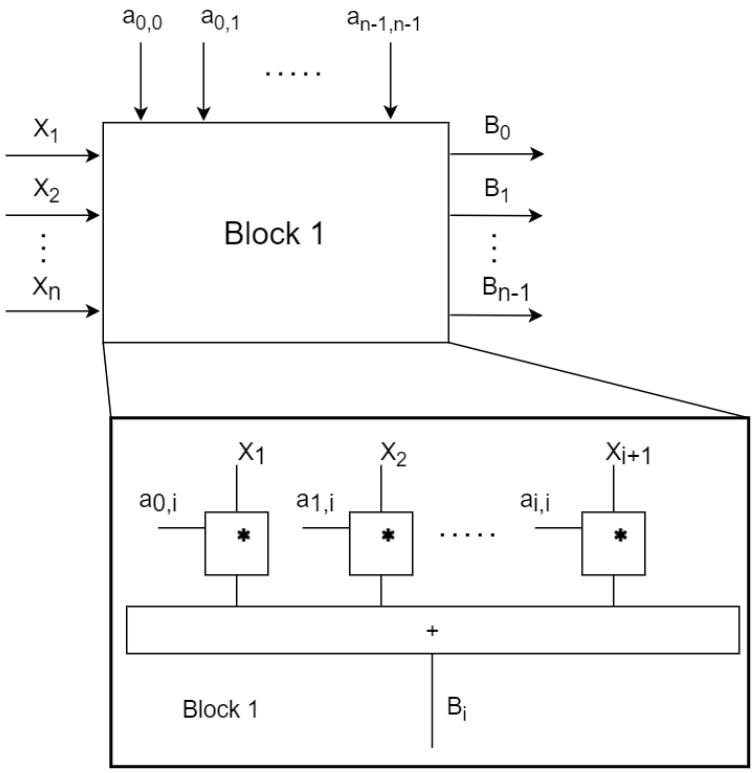


Figure 3.2: Structure for computation of the first-order radix values B_i

This characteristic matrix A can be pre-computed for a given moduli set and as the elements of A are bound by the moduli m_i, these elements are all small numbers. Thus, the RNS number $(x_1, x_2, \ldots, x_n)$ can be transformed into the first-order radix format $(x_0, x_1, \ldots, B_{n-1})$ by the means of small-range adders and multipliers as shown in Fig. 3.2. The decimal number

X can then be computed as:

$$X = [B_0 + B_1 m_1 + B_2 m_1 m_2 + \ldots + B_{n-1} m_1 m_2 \ldots m_{n-1}]_{m_1 m_2 \ldots m_{n-1} m_n}$$

This is the new CRT I that Wang proposed. The equation can be further reduced to

$$X = \big[B_0 + (B_1)_{m_2 \ldots m_{n-1} m_n} m_1 + (B_2)_{m_3 \ldots m_{n-1} m_n} m_1 m_2 + \ldots + (B_{n-1})_{m_n} m_1 m_2 \ldots m_{n-2} m_{n-1 m_1 m_2 \ldots m_{n-1} m_n}$$

using Proposition 1 and Fig. 3.3 shows a residue-to-binary converter implemented using the reduced equation. The benefit of the new CRT I as opposed to the traditional CRT is that the characteristic matrix A, as mentioned previously, is bound to the size of the moduli m_i and thus consist of relatively small numbers compared to the big numbers $\left|1/y_i\right|_{m_i}$ for traditional CRT, resulting in smaller addition and multiplication operations. This reduces the hardware overhead involved in the implementation of RNS-to-binary converters.

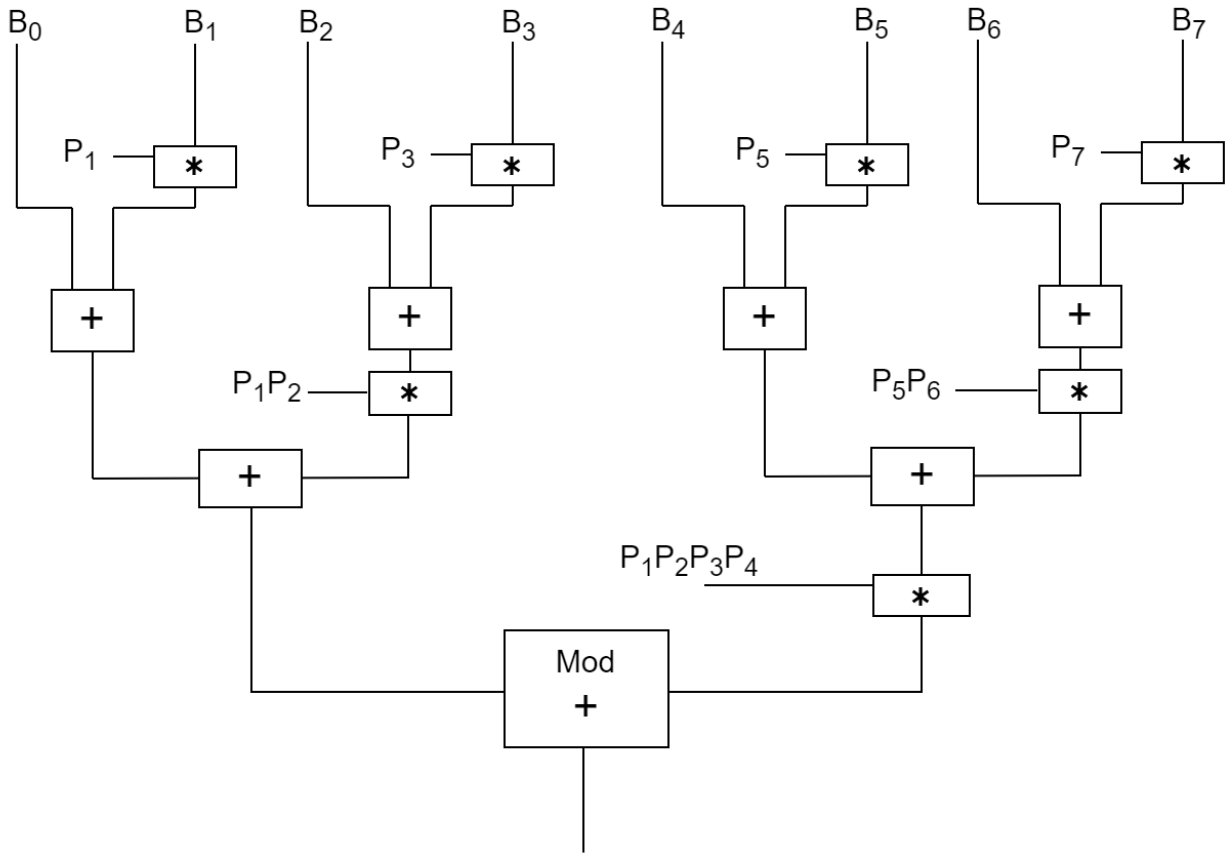


Figure 3.3: New CRT I based general RNS-to-binary converter

P(A)	3	4	5	6	7	8	9	10	11	12	13	14
A	7	5		31	9	127	17	11	23	13, 35, 39	57	43
		15			21		51	33	89	45, 65, 87		
					63		8	93		91, 105, 117		

Table 3.4: Table of moduli with $P(A) \leq 14$

Implementation

For the RNS-to-binary converter based on CRT-I implemented in this project the values for the characteristic matrix A was computed beforehand with the means of a MATLAB script. The computed characteristic matrix file was then interfaced as an LUT within the RNS-to-binary converter module. The computation of first-order radix vector B was achieved through the means of adders and multipliers as shown by Fig. 3.3 and period/half-period based residue generators were used for the calculation of the modulus to obtain the final binary output value.

3.2.2 Periodicity based Residue Generator

In the sequence of residues generated when modulus of the powers of two is taken w.r.t. to A , the periodicity of a given odd module A or the period of A is calculated as the smallest distance between two consecutive residue value of 1 obtained and represented as $P(A)$ (6). Hence,

$$P(A) = min\{i | i > 0 \,\text{and}\, |2^i|_A = 1\}$$

Table 3.4 lists the periodicity of some moduli $P(A) \leq 14$.

The periodicity property can be used to generalize the identity (7)

$$\left|2^{ta+j}\right|_{2^a-1} = \left|2^j\right|_{2^a-1}, \quad t \text{ is any nonnegative integer}$$

to

$$\left|2^{tP(A)+j}\right|_A = \left|2^j\right|_A, \quad t \text{ is any non-negative integer}$$

Then, for an n-bit number X which is divided into $d = \lceil n/P(A) \rceil$ $P(A)$-bit bytes B_i and b_i

HP(A)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
A	3	5	9	17	11	13	43	257	19, 27	25, 41	683	241
		15			13	65	129		57, 171	205	2049	4097
									513	1025		

Table 3.5: Table of moduli with $HP(A) \leq 14$

is the decimal value of B_i, the modulo of the number X w.r.t. A can be expressed as,

$$\begin{aligned} |X|_A &= \left| \sum_{i=0}^{d-1} b_i 2^{iP(A)} \right|_A \\ &= \left| \sum_{i=0}^{d-1} b_i \right|_A \end{aligned}$$

where $0 \leq b_i \leq A$.

Similarly, in the sequence of residues generated when modulus of the powers of two is taken w.r.t. to A, the half-periodicity of a given odd module A or the period of A, if it exists, is calculated as the minimum distance between two subsequent pairs of 1 and $A-1$ and represented as $HP(A)$. Hence,

$$HP(A) = min\{d | d > 0, d = j - i, |2^i|_A = 1 \text{ and } |2^j|_A = A - 1, 0 < i < \infty, 0 < j < \infty\}$$

The half-periods of some moduli having $HP(A) \leq 12$ have been displayed in Table 3.5.

While $P(A)$ exists for all A, $HP(A)$ only exists for some values of A. Half-period, as the name suggests, is called so because whenever it exists for a give A, it satisfies the following equation:

$$P(A) = 2 \cdot HP(A)$$

In the case where $HP(A)$ does not exist, its value is taken as infinity, i.e., $HP(A) = \infty$. The following recursive equation can be used to find $HP(A)$ and $P(A)$,

$$\left|2^i\right|_A = \left|2 \cdot \left|2^{i-1}\right|_A\right|_A$$

Then the half-periodicity property can be used to generalize the identity

$$\left|2^{t(a-1)+j}\right|_{2^{a-1}+1} = (-1)^t \left|2^j\right|_{2^a-1}, \quad t \text{ is any non-negative integer}$$

to

$$\left|2^{t\cdot HP(A)+j}\right|_A = (-1)^t \left|2^j\right|_A, \quad t \text{ is any non-negative integer}$$

Then, for an n-bit number X which is divided into $d = \lceil n/HP(A) \rceil$ $P(A)$-bit bytes B_i and b_i is the decimal value of B_i, the modulo of the number X w.r.t. A can be written as,

$$\begin{aligned} |X|_A &= \left| \sum_{i=0}^{d-1} b_i 2^{i\cdot HP(A)} \right|_A \\ &= \left| \sum_{i=0}^{d-1} (-1)^i b_i \right|_A \end{aligned}$$

where $0 \leq b_i \leq A$. As

$$\left|2^{HP(A)}\right|_A = -1$$

$$\begin{aligned} |-b_i|_A &= \left|\left(2^{HP(A)} - 1 - b_i\right) + \left(1 - 2^{HP(A)}\right)\right|_A \\ &= \left|\overline{b_i} + 2\right|_A \end{aligned}$$

as 1's complement of a given number b_i is $\overline{b_i} = 2^{HP(A)} - 1 - b_i$ where $\overline{b_i}$ represents the decimal value of the 1's complement or bit-wise complement of B_i.

Then the expression for the modulo of the number X w.r.t. A can be modified as,

$$|X|_A = \left| \sum_{i=0,even\, i}^{d-1} b_i + \sum_{i=0,odd\, i}^{d-1} \overline{b_i} + 2 \cdot \lfloor d/2 \rfloor \right|_A$$

where $2 \cdot \lfloor d/2 \rfloor$ is a correction for the $\lfloor d/2 \rfloor$ odd-numbered complemented bytes.

Implementation

The project involved construction of mod A generators using the property of periodicity and half-periodicity which were implemented as follows:

- MATLAB script was constructed for the computation of various parameters required such as value of d, bit-size of d, required bit-size of output, etc.

- For an n-bit binary input $(x_0, x_1, \ldots, x_{n-1})$, the bits were partitioned into d number of sub-binary numbers G_i,

$$G_i = x_k | k = tP(A) + j, 0 \leq j \leq P(A) - 1$$

.

- For period based residue generator, compute

$$sum = \sum_{i=0}^{d-1} G_i$$

.

- For half-period based residue generator, compute

$$sum = \sum_{i=0,even\, i}^{d-1} G_i + \sum_{i=0,odd\, i}^{d-1} \overline{G_i} + 2 \cdot \lfloor d/2 \rfloor$$

.

- Repeat step 3 or 4 until the bit-size of sum is less than or equal to $P(A) + 1$ or $HP(A) + 1$, for period and half-period based residue generators respectively. The number of repeated stages needed for a given $P(A)$ or $HP(A)$ can be found using a MATLAB script.

- Find residue of final reduced sum value using an LUT.

Chapter 4

Experimental Set-up and Results

4.1 Experimental Set-up

The testbench was developed using Xilinx Vivado Design Suite in Verilog, the output of which was written into a file and the results were verified using MathWorks MATLAB scripts which read the generated file and checked for discrepancies in the results. The modules were then synthesized using Cadence Encounter RTL Compiler to obtain the timing, area and power reports.

4.2 Results

The area, power and delay of the synthesized blocks have been tabulated in Tables 4.1, 4.2, 4.3. As observable, there is significant improvement in the area, power and delay of the converters implemented using the proposed residue generator and the new CRT-I as compared to the one using traditional CRT. This is attributed to the reduction of the inter-modulo operations used in the traditional CRT to multiplication and addition operations using the period based modulo generators as well as smaller values for which modulo computation is required for the new CRT-I method.

Residue Generator					
Synthesized Block	Area (λ^2)	No. of NAND-2 equivalent instances	Power (mW)	Delay (ns)	P $\times D$
Period based residue generator	124.90	87	0.006	2.34	0.014
Half-period based residue generator	144.36	101	0.007	2.39	0.016

Table 4.1: Synthesis results of implemented residue generator

Binary-to-RNS Converter					
Synthesized Block	Area (λ^2)	No. of NAND-2 equivalent instances	Power (mW)	Delay (ns)	P $\times D$
Traditional CRT	6660.4	4626	0.162	10.19	1.65
New CRT-I (32, 31, 21, 5)	499.7	347	0.03	2.92	0.085
New CRT-I (32, 17, 13, 11)	1440.7	1000	0.12	3.37	0.40

Table 4.2: Synthesis results of implemented binary-to-residue converters

RNS-to-Binary Converter					
Synthesized Block	Area (λ^2)	No. of NAND-2 equivalent instances	Power (mW)	Delay (ns)	P $\times D$
Traditional CRT	23870.5	16577	0.88	11.24	9.89
New CRT-I (32, 31, 21, 5)	9246.2	6421	0.32	7.88	2.54
New CRT-I (32, 17, 13, 11)	9686.8	6727	0.34	7.98	2.72

Table 4.3: Synthesis results of implemented residue-to-binary converters

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This thesis was aimed at solving the research problem of hardware efficient implementations for any given arbitrary moduli set. Although numerous literary works exist on efficient hardware models for special moduli set such as the $2^{k-1}, 2^k, 2^{k+1}$, such moduli are limited and the efficiency of such sets for the required range may be low leading to the need of effective arbitrary moduli set converters. With this objective in mind, the overall proposals made in this thesis have been encapsulated in this section.

This work demonstrates an area and power efficient binary-to-RNS-to-binary converter for arbitrary moduli set. The converter is based on the new Chinese Remainder Theorem I. Additionally residue generators for a given arbitrary module has been implemented by exploiting the number theoretic properties of an odd module due to its periodicity or, if it exists, its half-periodicity. The implemented model successfully reduces the difficulty in computation arising from the complicated inter-modulo operations required for back conversion from RNS domain to the binary domain into a simple architecture that uses an array of adders and multipliers to achieve the required modulo operations. The work also reduces the need for multiple large-size LUTs as the periodicity property has been used to limit the size of the resultant value whose moduli is to be found.

5.2 Future Research Directions

On the basis of the research results presented in this thesis, future work can be done on points identified below:

- The implemented arbitrary residue generator suffers from differences in adder width and depth due to the variation in the periodicities of different moduli. In the implementation the size, depth and number of required adders was calculated using a MATLAB script which generated the required constants for the Verilog file, however, using properties such as leveraging the distributive property instead of periodicity might help limit the width of the required adder.

- As embedded digital signal processing is computationally dominated by discrete convolution operations, it requires hardware having high speed and low power capacity

for the interleaved addition and multiplication operations involved in discrete convolution. For hardware optimization of existing digital signal processing algorithms, RNS domain based arithmetic circuitry can be designed which can then be used in conjuction with the implemented converters to obtain an RNS based DSP processor.

- Neural networks heavily employ MAC-operations (multiply-and-add) for the computation of convolution and thus can also benefit from the implemented converters.

Bibliography

[1] R. Chokshi, K. S. Berezowski, A. Shrivastava, and S. J. Piestrak, "Exploiting residue number system for power-efficient digital signal processing in embedded processors," in *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, (New York, NY, USA), p. 19–28, Association for Computing Machinery, 2009.

[2] A. Hiasat, "A residue-to-binary converter for the extended four-moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, pp. 2188–2192, 2017.

[3] K. V. Vardhan, G. Sailakshmi, S. Musala, and A. Srinivasulu, "Analysis and design of high speed residue to binary reverse converter for the moduli set 2n+ 1, 2n,2n-1," in *2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pp. 53–56, 2020.

[4] P. Patronik, "On reverse converters for arbitrary multi-moduli rns," *Integration*, vol. 75, pp. 158–167, 2020.

[5] Y. Wang, "Residue-to-binary converters based on new chinese remainder theorems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 3, pp. 197–205, 2000.

[6] S. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Transactions on Computers*, vol. 43, no. 1, pp. 68–77, 1994.

[7] A. Avizienis, "Arithmetic error codes: Cost and effectiveness studies for application in digital system design," *IEEE Transactions on Computers*, vol. C-20, no. 11, pp. 1322–1331, 1971.