Effective Reverse Conversion in Residue Number System Processors

Kazeem Alagbe Gbolagade

Effective Reverse Conversion in Residue Number System Processors

---

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen

op dinsdag 21 december 2010 om 10:00 uur

door

Kazeem Alagbe GBOLAGADE

Master of Science in Computer Science
Department of Computer Science
University of Ibadan
Nigeria.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. H.J. Sips

Copromotor:
Dr. S. Cotofana

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter	Technische Universiteit Delft
Prof. dr. ir. H.J. Sips, promotor	Technische Universiteit Delft
Dr. S. Cotofana, copromotor	Technische Universiteit Delft
Prof. dr. J.M. Muller	LIP, Ecole Normale Superieure, Lyon, France
Prof. dr. K.S. Nokoe	University for Development Studies, Ghana
Prof. dr. P. French	Technische Universiteit Delft
Dr. E.A. Suarez	Universidade de Santiago de Compostela, Spain
Dr. L.A. Sousa	Universidade Tećnica de Lisboa
Prof. dr. J.R. Long	Technische Universiteit Delft, reservelid

ISBN 978-90-72298-10-2

Keywords: Residue Number Systems, Data Conversion, Chinese Remainder Theorem, Mixed Radix Conversion, Digital Signal Processing.

Printed in The Netherlands

This dissertation is dedicated to all the members of my family.

Effective Reverse Conversion in Residue Number System Processors

Kazeem Alagbe Gbolagade

Abstract

In this dissertation, we propose effective Residue Number System (RNS) to Weighted Number System conversion techniques. This research follows the two traditional conversion methods: the Mixed Radix Conversion (MRC) and the Chinese Remainder Theorem (CRT). In the first line of research, we investigate two MRC based techniques with k being the number of moduli. First, we introduce an RNS to MRC technique, which addresses the computation of Mixed Radix Digits in such away that enables the MRC parallelization. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$. Second, we generalize a previously proposed technique that was restricted to 5-moduli set such that it can be utilized in conjunction with any RNS with the set of relatively prime integer moduli $\{m_1, m_2, m_3, ..., m_k\}$. Just like the first scheme, this second technique also results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$. In the second line of research, we propose a number of efficient converters based on the simplification of the traditional CRT. First, we assume a general $\{m_1, m_2, m_3, ..., m_k\}$ moduli set, where $m_1 > m_2 > m_3 > ... > m_k$, with the dynamic range $M = \prod_{i=1}^{k} m_i$ and introduce a modified CRT that requires mod-m_k instead of mod-M calculations. Subsequently, we further simplify the conversion process by focusing on moduli sets with common factors, i.e., $\{2n+2, 2n+1, 2n\}$ and $\{2n+3, 2n+2, 2n+1, 2n\}$. Additionally, for the moduli set $\{2n+2, 2n+1, 2n\}$, we propose further simplifications which result in a scheme that does not even require explicit modulo operation computation. Second, for the $\{2n+1, 2n, 2n-1\}$ moduli set, we propose a novel converter, which also doesn't require explicit modulo operation computation during the conversion processes. Third, we propose two efficient memoryless reverse converters for the $\{2^{n+1}-1, 2^n, 2^n-1\}$ moduli set. Fourth, we propose two efficient adder based reverse converters for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set. Finally, two CRT and one MRC based reverse converters are proposed for the moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. Experimental results indicate that our proposals substantially outperform state of the art equivalent converters in terms of area, delay, and power consumption.

Acknowledgements

First of all, I would like to thank late Prof. dr. Stamatis Vassiliadis, who was and remains the patron of the Computer Engineering Laboratory group, for signing my admission letter. Death unfortunately did not permit me to enjoy friendship with Stamatis. May his gentle soul rest in perfect peace. I'm also thankful to Prof. dr. Henk Sips for serving as my promotor and for his efficient ways of handling issues.

Next, I especially would like to express my deepest gratitude to my advisor, Prof. Sorin Cotofana, who is a great mentor that has inspired me to be a better person both in life and science. I have learnt a lot from his belief that doing a PhD is synonymous to building a house in which a very strong foundation is needed. I thank him for investing a lot of time to ensure that we had a good starting point. Additionally, he exposed me to critical analysis of scientific issues, the art of neat technical writing, and has contributed in countless ways to the successful completion of this dissertation. I'm very much grateful for all the time he shared with me at work and also for everything he has done for me.

I'm very much grateful to the examination committee for the useful feedbacks and comments on the dissertation. Particularly, I would like to thank Prof. dr. Jean-Michel Muller for his very detailed comments.

I wish to say a big thank you to Prof. Leonel Sousa for providing grant for me to visit his group in TU Lisbon in Portugal. I really appreciate all the precious time he shared with me in long talks on various aspects of Residue Number Systems, theory and applications. He is a great scientist and multivator who always provides energy to his students to do more work. I've learnt a lot from his hardworking and humble type of life. Thank you very much, Leonel. I wish also to thank Dr. Ricardo Chaves for his immense contributions towards the development of this work. Although, he was initially a colleague, he acted as my daily advisor during my visit at IST in TU Lisbon and I have benefited a lot

from his rich experience on digital computer arithmetic. Thank you, Ricardo.

I would like also to thank Professors Koen Bertels and Said Hamdioui and the rest of the professors in the group for always ready to help and for their constant encouragements.

I would like to specially thank the following people: George Razvan Voicu, Yao Wang, Sandra Irobi, Ayo Imoru, Austine, Innocent Agbo, Zaidi Haron, Seyab Khan, Christophorous Chakiris, Cristos, Laiq Hassan, Hassan Shabi, Zubair, Fakhar, Faisal, Tariq Abdulai. George Razvan Voicu gets an extra acknowledgement for criticising one of my theorems during his M.Sc days. His criticism has greatly improved the quality of this work. Many thanks George. I would like to specially thank Muttaqiallah Taouil and Saleh for translating my abstract and propositions into Dutch.

I'm very much grateful to Prof. dr. Sagary Nokoe, the Vice-Chancellor of the University for Development Studies (UDS), Ghana, for his moral, administrative, and academic support. He has been a great mentor and a multivator. I wish to also express my sincere appreciation to Dr. Elkanah Oyetunji, the Dean of the faculty of Mathematical Sciences, UDS, Ghana, for his constant advice and encouragement in more difficult moments. I would also like to acknowledge the following people: Prof. David Millar, Dr. Kenneth Pelig-Ba, and all other staff of UDS, too numerous to mention, that have helped me one way or the other .

I'm thankful to my parents Prof. dr. Wasiu Alagbe Gbolagade, late Mrs. Kafilat Alagbe (may her gentle soul rest in perfect peace), and to all the members of my family, for all their numerous support.

I would specially like to thank my dear wife and children for their unlimited support and understanding. They took the pain to stay without my presence in so many occassions.

Finally, I am thankful to Bert, Erick, Eef, Monique, and Lidwina Tromp for their technical and administrative support in the Computer Engineering Laboratory. Lidwina gets an extra acknowledgement for helping me with the bureaucratic processes I came across.

Kazeem Gbolagade Delft, The Netherlands, 2010

Contents

List of Tables

List of Figures

List of Acronyms

WNS	Weighted Number System
ASIC	Application-Specific Integrated Circuit
GCD	Greatest Common Divisor
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
LSB	Least Significant Bit
MSB	Most Significant Bit
RNS	Residue Number System
CRT	Chinese Remainder Theorem
MRC	Mixed Radix Conversion
RRNS	Redundant Residue Number System
QRNS	Quadratic Residue Number System
DCT	Discrete Cosine Transform
FCT	Fast Cosine Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
MC	Multicarrier
CDMA	Code Division Multiple Access
FPGA	Field Programmable Gate Array
ROM	Read Only Memory
MRD	Mixed Radix Digit
GCM	Greatest Common Measure
HCF	Highest Common Factor
GCF	Greatest Common Factor
MR	Mixed Radix
MCRT	Modified Chinese Remainder Theorem
MATR	Matrix Method
CMOS	Complementary Meta Oxide Semiconductor
VHDL	VHSIC Hardware Description Language

AT	Area-Time
CSA	Carry Save Adder
CPA	Carry Propagate Adder
EAC	End Around Carry
IMRC	Improved Mixed Radix Conversion
DE	Diophantine Equation
LCM	Lowest Common Multiple

Chapter 1

Introduction

In the last decades, we have witnessed an ongoing increase in integrated circuit performance mainly due to advances in fabrication technology and improvements in computational paradigms [62]. The major challenge in improving performance from the computational paradigm point of view is the reduction/elimination of the carry propagation chains inherent to Weighted Number Systems (WNS), e.g., binary number systems, decimal number systems. While this is an intrinsic performance limiter for arithmetic units and processors built based on WNS, several attempts have been made to overcome the speed limitations by following two main research avenues as follows:

The carry propagation through the conventional ripple-carry adders, which is the main contributor to the addition delay, can be accelerated by using fast addition techniques [143]. Those make use of specialized circuitry able to deserialize the carries calculation via methods like carry look ahead, carry-skip, prefix calculation, anticipated calculation, etc. These fast addition techniques are very important in improving arithmetic units performance because other arithmetic operations such as multiplication and division are based on addition, thus their delay heavily depends on the addition delay. For non redundant number systems, e.g., traditional binary and decimal number systems, the delay of such fast adders is logarithmically bounded by the number of operand digits [111].

An alternative way to speed up the addition process is to make use of number system with specialized carry characteristics, i.e., proposing alternative number representation systems, e.g., Redundant Signed Digit (RSD) number representation systems [18, 28, 63, 69, 93, 94, 103] and Residue Number Systems (RNS) [94].

In RNS, a set of moduli which are all independent (relatively prime) of one another are given. In this context, integer operators are represented by the residues (remainders) of each modulus and arithmetic operations are individually based on those residues [77]. The basis for an RNS is a set of relatively prime integers $\{m_1, m_2, m_3, ..., m_k\}$ such that $gcd\,(m_i, m_j) = 1$ for $i \neq j$, where gcd means the greatest common divisor of m_i and m_j. For such a system, $M = \Pi_{i=1}^{k} m_i$ is the dynamic range and any integer $X \in [0, M-1]$ can be uniquely represented as $X = (x_1, x_2, x_3, ..., x_k)$, where $x_i = |X|_{m_i}$, $0 \leq x_i < m_i$. For convenience sake, in this thesis, $|X|_{m_i}$ denotes the X mod m_i operation.

The sought characteristic of such alternative number systems is the capability to provide support for carry-free addition as this directly results in high-speed arithmetic units. The most important property of RSD number representation systems is the possibility of performing carry-free addition and borrow-free subtraction [93]. Redundant number systems provide the possibility of addition with sub-logarithmic and even constant latency [63]. Therefore, extremely fast constant-time addition can be performed with redundant number systems that support carry-free addition [93].

However, addition/subtraction in RSD requires limited carry since the carries are saved or stored and carry propagation delay occurs once at the very end of the computation [94]. Although algorithm for performing carry free addition is available, it is only valid if certain conditions are satisfied [94]. Thus, we seek a number system, which generally supports carry free operations. The RNS [37, 111, 113] is such an integer system exhibiting the capabilities to support parallel, carry-free addition, borrow-free subtraction, and digit by digit multiplication. Moreover, it provides support for fault tolerance [11, 51, 137], which is becoming a crucial aspect as it is getting more and more expensive/difficult to fabricate perfect (predictable) devices in the context of deep sub-micron fabrication technologies [62].

Thus far, most of the RNS related research has been done in the utilization of RNS in Digital Signal Processing (DSP) applications, e.g., Digital Filtering [31, 64, 65, 90, 118, 136], Convolutions, Correlations, Discrete Cosine Transform (DCT) [34, 35], Discrete Fourier Transform (DFT) [58, 114, 116, 119], Fast Fourier Transform (FFT) [31, 80, 108, 117]. Additionally, RNS has also been applied in low power design [19,61,66,73,91,92,94,95,110], number theory [10, 74, 104], and digital communications [78, 79, 105, 138].

Despite all these desirable RNS features, it has not found a widespread usage in general purpose processors as the following RNS challenges must be

properly addressed in order to be able to design RNS based general purpose architectures: sign detection, magnitude comparison [4, 15, 33, 67, 115, 132], overflow detection, moduli selection, residue to binary/decimal conversion and vice-versa, division [9, 76], and other complex arithmetic operations.

Effective data converters are required in order to efficiently implement the difficult RNS operations, e.g., magnitude comparison, sign detection, and division, and also to build fault tolerant RNS architectures. Thus, for general purpose RNS processors to become a reality, high-speed data converters are required. Due to these reasons, in this thesis, we propose efficient conversion techniques stemming from either the Mixed Radix Conversion (MRC) or the Chinese Remainder Theorem (CRT).

The remainder of this chapter is organized as follows. In Section 1.1, we present basic background information on RNS specific data conversion techniques. Section 1.2 discusses related research work and the research issues that are addressed in this thesis, while in Section 1.3, this introductory chapter is concluded with the discussion of thesis overview and how the developed research work is structured and presented.

1.1 Data Conversion Background

The origin of RNS can be traced to the puzzle given by Sun Tzu [108], a Chinese Mathematician and is illustrated as follows. How can we determine a number that has the remainders 2, 3, and 2 when divided by the numbers 7, 5, and 3, respectively? This puzzle, written in the form of a verse in the third century book, Suan-ching by the Chinese scholar Sun Tzu, is perhaps the first documented use of number representation using multiple remainders. The answer to this puzzle, 23, is outlined in Sun Tzu's historic work. The puzzle essentially asks us to convert the RNS number $(2|3|2)_{RNS(7|5|3)}$ into its decimal equivalent. Sun Tzu formulated a method for manipulating these remainders (also known as residues), into integers. This method is regarded today as the Chinese Remainder Theorem (CRT). The CRT, as well as the theory of RNS, was set forth in the 19th century by Carl Friedrich Gauss in his celebrated Disquisitiones Arithmetical [108]. Lehmer, Svoboda, and Valach first built hardware using RNS. This was reported in 1955 [108].

Considerable work was done at various laboratories on building hardware based on RNS during 1950's and 1960's. Szabo and Tanaka [111] and Watson and Hastings [134] documented the results in this area in their books in 1967. Between 1967 and 1977, there was a setback in RNS research out-

put as the RNS researchers and processor designers were sceptical about the dream of RNS processors coming to reality. In 1977, Jenkin and Leon [65] rekindled the interests of researchers in this area by their pioneering work. Interesting results were made available by many researchers between 1977 and 1985 [9, 10, 59, 64, 67, 112, 122]. The research work carried out in this area up to 1986 was compiled in [108]. Mohan [83] continued with the trend and reported a collection of articles up to 2001. From 2002 to the present time, interesting results have been reported and we take note of the efforts of the following researchers among others [38, 40, 43, 44, 72, 101, 102, 109, 129].

RNS architectures are typically composed of three main parts, namely, a binary-to-residue converter, residue arithmetic units, and a residue-to-binary converter. The residue-to-binary converter is the most challenging part of any RNS architecture. For RNS design to be competitive, the conversion process must be very fast as otherwise the RNS fast arithmetic advantage may be nullified by the slow conversion processes. As those conversions are computation demanding and have crucial influence on the speed and complexity of the RNS architectures, designing efficient converters has been an important task in the realization of different RNS based applications and processors and several proposals have been reported up to date [1,7,8,15,27,47,83,87,107,108,112,122].

Most of the work on residue to binary conversion is based either on the CRT [24, 27, 41, 44, 45, 55, 56, 72, 87, 97, 123–125] or on the MRC [3, 42, 46, 48, 59, 81, 89]. CRT is desirable because the computation can be parallelized while MRC is by its very nature a sequential process. However many up to date RNS to binary/decimal converters are based on MRC due to the complex and slow modulo-M operation (M is the dynamic range of the system, and thus a rather large constant) required by CRT.

The traditional CRT is defined as follows. For a moduli set $\{m_1, m_2, m_3, ..., m_k\}$ with the dynamic range $M = \prod_{i=1}^{k} m_i$, the residue number $(x_1, x_2, x_3, ..., x_k)$ can be converted into the decimal number X, as follows [111]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M , \tag{1.1}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

The main drawback of CRT emerges from the required modulo-M operation which, given that M is a rather large number, this operation can be time consuming and rather expensive in terms of area end energy consumption. The

MRC is an alternative method which does not involve the large modulo-M calculations. Given an RNS number $(x_1, x_2, x_3, ..., x_k)$ for the moduli set $\{m_1, m_2, m_3, ..., m_k\}$, the decimal equivalent of it can be computed as [111]:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 + ... + a_n m_1 m_2 m_3 ... m_{k-1} \tag{1.2}$$

where the Mixed Radix Digits (MRDs) $a_{i,i=1,k}$, can be computed as:

$$\begin{aligned}
a_1 &= x_1 \\
a_2 &= \left| (x_2 - a_1) \left| m_1^{-1} \right|_{m_2} \right|_{m_2} \\
a_3 &= \left| \left((x_3 - a_1) \left| m_1^{-1} \right|_{m_3} - a_2 \right) \left| m_2^{-1} \right|_{m_3} \right|_{m_3} \\
&\cdot \\
&\cdot \\
a_k &= \Big| (((x_k - a_1) \left| m_1^{-1} \right|_{m_k} - a_2) \left| m_2^{-1} \right|_{m_k} - ... \\
&\quad -a_{k-1}) \left| m_{k-1}^{-1} \right|_{m_k} \Big|_{m_k}
\end{aligned} \tag{1.3}$$

For the MRDs a_i, $0 \leq a_i < m_i$, any positive number in the interval $[0, \Pi_{i=1}^{k} m_i - 1]$ can be uniquely represented. The major MRC advantage, as can be seen from Equation (1.3) is that the calculation of $a_i, i = 1, k$ can be done only using arithmetic mod-m_i unlike CRT, which involves arithmetic mod-M, M being the system dynamic range, a rather large constant.

We note inhere that Equations (1.1) and (1.2) can be directly utilized, only if the moduli set $\{m_1, m_2, m_3, ..., m_k\}$ are relatively prime and that Euclidean algorithm [29, 135] is the common way to verify this, i.e., if gcd$(m_i, m_j) = 1$, for $i \neq j$.

1.2 Related Work and Problem Statement

As discussed earlier, the existing reverse converters are stemming from either the MRC or the CRT. Thus, all the discussions here follow the two techniques starting with MRC. The previously proposed fast MRC techniques either make use of look up tables [3, 59, 67, 68, 81, 89, 111], or reduce the number of arithmetic operations that are involved in the conversion processes [71, 139–141]. The later approach is desirable because a suitable reduction method can be utilized in conjunction with appropriate selection of moduli, which can produce

unity for all the required multiplicative inverses in the conversion processes. This can lead to a fast and memoryless reverse converter.

MRC based techniques, which considerably reduce the number of involved arithmetic operations during the conversion processes have been presented in [139–141]. It can be inferred from the description given in Equation (1.3) and in line with the discussion in [141] that a total of $\frac{k(k-1)}{2}$ arithmetic subtractions and multiplications are required in order to compute the MRDs for k-moduli sets. This means that the conversion process that computes the MRDs for an RNS with k moduli set has an asymptotic complexity in the order of $O(k^2)$.

The improved MRC described in [141] requires a total of $\frac{k(k-1)}{2}$ subtractions just as in [111] but can reduce the arithmetic multiplications to $(k-2)$ instead of $\frac{k(k-1)}{2}$ required in [111] for computing the MRDs. Even though the number of arithmetic multiplications was reduced, the asymptotic complexity was not improved beyond $O(k^2)$. We therefore first address the following research question:

- Can we reduce the asymptotic complexity such that we obtain a linear complexity, i.e., $O(k)$?

Also in the MRC line of thinking, it is worth mentioning the matrix method described in [140] for the particular moduli set $\{11, 7, 5, 3, 2\}$. In this approach 4 modular equations have to be solved, which results in a quite effective way to perform the RNS to WNS conversion for the assumed moduli set. In view of this, in this thesis, we address the following open question:

- Given that there exists an unrestricted 5 moduli MRC based Matrix method, can we generalize the matrix method such that it becomes applicable for any RNS with relatively prime integer moduli $\{m_1, m_2, m_3, ..., m_k\}$?

As previously mentioned, the other research avenue followed by the community is based on CRT. While for MRC, the discussion can be done in general, CRT based converters have structures and performance which heavily depend on the moduli set they target. In view of that, the rest of the discussion in this section is organized per type/class of moduli. The large majority of available high-speed CRT based reverse converter architectures are based on the triple moduli set $\{2^n+1, 2^n, 2^n-1\}$ as they can exploit the useful properties of the numbers (2^k-1) or (2^k+1). However, the moduli sets $\{2n+1, 2n, 2n-1\}$ and $\{2n+2, 2n+1, 2n\}$ are also considered useful because the numbers

are consecutive, enabling nearly equal width adders and multipliers in the hardware implementation [2]. For the moduli sets $\{2n+2, 2n+1, 2n\}$ and $\{2n+1, 2n, 2n-1\}$, CRT based RNS to decimal converters, which "do not require the explicit computations of modulo operations" (this expression implies that, contrary to the conventional traditional CRT, the modulo operation computation is required only in some cases and can even be completely eliminated) have been presented in [2, 99, 100]. This can be seen as an interesting breakthrough as the major CRT problem is eliminated but despite the fact that no explicit modulo operations are required by these techniques, the numbers involved in the computations are large. Thus, by implication, they are rather slow and their hardware implementations require large area. This leads to the following research questions:

- For each of the moduli sets $\{2n+2, 2n+1, 2n\}$ and $\{2n+1, 2n, 2n-1\}$, can the modulo operation be reduced to minimum? Can the modulo operation computations be completely eliminated?
- For applications requiring larger dynamic range than the one offered by $\{2n+2, 2n+1, 2n\}$, can this moduli set be extended to $\{2n+3, 2n+2, 2n+1, 2n\}$? What are the implications of this extension in terms of conversion cost?

On the other hand, given that modulo arithmetic for the moduli set $\{2^n+1, 2^n, 2^n-1\}$ is relatively easy, building efficient reverse converters for this moduli set is of practical interest. Andraos and Ahmad [6] introduced innovative concepts in reverse converter design. They derived closed form expressions for the moduli inverses so as to reduce the conversion complexity. Also, a general technique of dividing both sides of the CRT expression by 2^n and then taking its modulus with respect to the product of the remaining moduli was introduced. This reduces the adder size by n bits. Novel reverse converters have also been proposed in [32, 60].

In [120], a faster reverse converter was presented together with magnitude comparisons of two RNS numbers. Later on, a faster and cheaper implementation was accomplished by Piestrak [96]. The reverse converter proposed in [60] depends on simple mathematical relationships without using MRC or CRT. The technique mainly consists of two steps, the first step is to find a number that corresponds to the first and last moduli $(2^n-1, 2^n+1)$ and the second step is to combine this number with the modulus 2^n. A similar approach was adopted by Bi and Jones [14]. In [14], CRT was used to combine

the first and the last moduli and finally in order to combine the result with the second one, MRC was employed. Piestrak [96]'s work was modified in [12] resulting into a high-speed and area efficient reverse converter.

A new property of a CRT decomposition was presented in [30]. This property is applicable to any RNS moduli set and has been applied to the moduli set $\{2^n + 1, 2^n, 2^n - 1\}$ in order to obtain an area-delay efficient reverse converter. Part of the total dynamic range is unusable due to the CRT decomposition.

In [133], new and uniform algorithms were designed using the New CRT for the RNS to binary conversion. Three different converters using either $2n$-bit or n-bit adders were proposed. The $2n$-bit adder-based converter is faster and requires about half the hardware required by the previous methods [6, 60, 96]. For the n-bit adder-based implementations, one of the converters is twice as fast as the previous method [36] with similar amount of hardware resources, whereas another n-bit converter achieves improvement in both speed and area. For the n-bit adder-based converter, the amount of hardware is similar compared with the one in [30]. However, as said before, in [30], not the entire dynamic range of numbers is used.

In [16], new theorems, which further reduces the modulo operation required by the modified CRT (New CRT) for three moduli sets was proposed. The modulo part of the Residue to Binary converter for the moduli set $\{2^n + 1, 2^n, 2^n - 1\}$ based on the proposed theorems was also presented. The FPGA implementation results show that the proposed converter is about twice faster and requires 50% less hardware and power for the modulo operation than the converter in [133]. However, this proposal does not give a complete residue-to-binary converter. It only considers the modulo part delay, which is not the only factor that determines the delay of a complete residue-to-binary converter. Thus, a more efficient converter is still desirable. In [52, 84, 86, 87], efficient RNS reverse converter structures were proposed. In [86], an efficient reverse converter was proposed for the moduli set $\left\{2^{n+1} + 1, 2^n, 2^{n+1} - 1\right\}$, which is also an enhancement of the traditional moduli set $\{2^n + 1, 2^n, 2^n - 1\}$.

However, for applications requiring larger dynamic range, the traditional moduli set has been extended to four, five, or multi-moduli sets [5, 7, 13, 20–23, 55, 72, 82, 84, 106, 121, 142]. As discussed earlier, the moduli set choice and the conversion algorithm selection greatly influence the area and delay of the resulting RNS to binary converter design. Currently, the following algorithms are available: The traditional CRT and MRC [111], the core function [142], and the New CRT [131]. The New CRT [131] eliminates the drawbacks of

both the traditional CRT and MRC algorithms, resulting in notable improvement in the conversion algorithms. However, the applicability of the New CRT [131] depends on the chosen moduli set. Due to this fact, some of the recently proposed efficient reverse converters are still based on the traditional approaches. The reverse conversion algorithms presented in [5, 20, 106, 142] are based on the New CRT [131].

The reverse conversion algorithm presented in [5] is for the moduli set $\{2^{n+1}+2^n-1, 2^n+2^{n-1}-1, 2^n-1, 2\}$, which is having a limited dynamic range because the first modulus is a constant, i.e., 2. Another notable reverse converter based on the New CRT [131], presented in [106] for the moduli set $\{2^n+3, 2^n+1, 2^n-1, 2^n-3\}$, has multiplicative inverses which are in poor formats. Additionally, three ROM tables, which are very expensive for large dynamic range were used to eliminate the modular multiplications. For the moduli set $\{2^n-3, 2^n+1, 2^n-1, 2^n+3\}$, Mohan [84] suggested a ROM-less implementation using Montgomery's modulo multiplication in order to realize the conversion. This avoids the expensive ROM tables utilized in [106]. Recently, a memoryless reverse converter was proposed for another new moduli set $\{2^{2n+1}-3, 2^{2n}-2, 2^n+1, 2^n-1\}$ [142]. This moduli set is unbalanced and the resulting structure is very complex with also higher area cost and conversion delay compared to the dynamic range of $6n$-bits it provides. Cao, et al [21] presented an efficient RNS to Binary converter for the moduli set $\{2^{2n}+1, 2^n+1, 2^n, 2^n-1\}$, which is an extension of the traditional moduli set $\{2^n+1, 2^n, 2^n-1\}$. The choice of the fourth modulus $2^{2n}+1$ was quite innovative over other previous choices, i.e., $2^{n+1}+1$ in $\{2^{n+1}+1, 2^n+1, 2^n, 2^n-1\}$ [13] and $2^{n+1}-1$ in $\{2^{n+1}-1, 2^n+1, 2^n, 2^n-1\}$ [121]. The reverse converter proposed in [21] is composed of only 2-levels of carry save adders and a modulo-$2^{4n}-1$ adder. However, in the moduli set $\{2^{2n}+1, 2^n+1, 2^n, 2^n-1, \}$, the increase of the word length of one of the moduli to $2n$ bits as in the case of the moduli set $\{2^n+1, 2^n, 2^n-1\}$ increases the computation time of all the operations of the processor corresponding to the modulus $2^{2n}+1$. Thus, there is a need to examine whether this can be taken into account in considering modified three moduli set $\{2^{2n}+1, 2^{2n}-1, 2^n\}$ in place of four moduli set $\{2^{2n}+1, 2^n+1, 2^n, 2^n-1\}$.

The moduli set $\{2^{2n}+1, 2^n, 2^{2n}-1\}$ was independently investigated in [52] and [84] and the same results were reported. With the same dynamic range, the reverse converter structure in [52, 84] is composed of only one CSA and a modulo-$2^{4n}-1$ adder producing better results in terms of area and delay when compared with the reverse converter in [21]. A generalized reverse converter

structure for the moduli set $\{2^n+1, 2^n, 2^n-1\}$, which is also a generalized form of the reverse converters in [52, 84], was presented in [87]. In general, a reverse converter based on the moduli set of the form $\left\{2^b+1, 2^a, 2^b-1\right\}$ will require only one $2^{2b}-1$-bit CSA and a $2^{2b}-1$-bit modulo adder.

The main disadvantage of the popular triple moduli set described above, which serves as an obstacle from obtaining a higher speed-up, is that the third modulus, 2^n+1, requires $(n+1)$ bits to represent 2^n+1 states. That is, many parts of the states remain unused. The differences in the complexity of the arithmetic units for the various moduli of the set are not negligible (the imbalance between the elements of the moduli set), in particular, the longer critical path of the 2^n+1 moduli set. Moreover, in spite of just exceeding by one the power of 2, the additional bit of the moduli 2^n+1 multiplication originates a more complex computational structure and consequently the processing time increases, when compared with the other moduli multiplications. Additionally, arithmetic operations with respect to 2^n+1 is not as simple as left circular rotation in a 2^n-1 modulus. These stated issues usually degrade the performance of the entire RNS architecture. For the imbalance problem, a modification of the moduli set $\{2^n+1, 2^n, 2^n-1\}$, which corresponds to the overloading of the binary base element (channel) has been proposed in [25–27] in order to obtain more balanced structures.

On the other hand, since arithmetic operations with respect to 2^n-1 modulus is simpler and faster than 2^n+1 modulus, alternative moduli sets, which are free of 2^n+1-type arithmetic have been proposed in [57, 75, 85, 88, 127, 128]. The first step towards proposing alternative moduli set was taken in [57] by proposing the moduli set $\left\{2^n, 2^n-1, 2^{n-1}-1\right\}$ together with its adder-based reverse converter. Later on, a similar moduli set $\left\{2^{n+1}-1, 2^n, 2^n-1\right\}$, which was enhanced in [88], was proposed in [85]. A reverse converter with better Area-Time (AT) requirements than the one in [85] was presented in [75]. Given that the required area cost for the reverse converter in [75] is large, we provide solutions to the following research question:

- Can we obtain a reverse converter with better area cost with small or no penalty in the conversion time when compared to the one in [75]?

As said earlier, the moduli set $\left\{2^{n+1}-1, 2^n, 2^n-1\right\}$ was enhanced to $\left\{2^{2n+1}-1, 2^n, 2^n-1\right\}$ in [88]. A MRC based reverse converter, which requires a high conversion time has been proposed for this set. We resolve the following issue:

- Can we obtain a reverse converter with better conversion time with small

or no penalty in the area cost when compared to the proposed reverse converter in [88]?

Given that in the moduli set $\{2^{2n+1}-1, 2^n, 2^n-1\}$, the binary channel is underutilized and that applications requiring larger dynamic range are of practical interest, we present solutions to the following question:

- Can we obtain an efficient reverse converter for an extended moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$?

In this thesis, we address all the previously stated research questions and propose theoretical solutions which are also evaluated by means of VHDL implementations.

1.3 Main Results and Thesis Overview

In this thesis, we propose several novel reverse converters based on either the traditional CRT or MRC. The thesis organization and its main contributions are as follows:

- Chapter 2 introduces an RNS to MRC technique, which addresses the computation of Mixed Radix Digits (MRDs) in such a way that enables the MRC parallelization. Given an RNS with the set of relatively prime integer moduli $\{m_1, m_2, m_3, ..., m_k\}$, the key idea behind the proposed technique is to maximize the utilization of the modulo-m_i adders and multipliers embedded in the RNS processor functional units. For an n-digit RNS number $X = (x_1, x_2, x_3, ..., x_k)$ the method requires k iterations. However, at iteration i, the modulo-m_i units are utilized for the calculation of the MRD a_i, while the other modulo units are calculating intermediate results required in further iterations. This approach results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$ while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(k^2\right)$, k being the number of moduli.

- In Chapter 3, we present a matrix based method that is a generalization of a previously proposed 5-moduli set restricted technique such that it becomes applicable to any RNS with the set of relatively prime integer

moduli $\{m_1, m_2, m_3, ..., m_k\}$. Next, we simplify the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers which are part of the RNS processor functional units. With higher reduction in the number of arithmetic operations than the ones in Chapter 2, this scheme also results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$.

- Chapter 4 investigates moduli sets with common factors. First, we assume a 3-moduli set $\{m_1, m_2, m_3\}$, where $m_1 > m_2 > m_3$, with the dynamic range $M = \prod_{i=1}^{3} m_i$ and introduce a modified CRT that requires mod-m_3 instead of mod-M calculations. Subsequently, we further simplify the conversion process by focussing on $\{2n+2, 2n+1, 2n\}$ moduli set, which has a common factor of 2. We introduce in a formal way a CRT based approach for this case, which requires the conversion of the $\{2n+2, 2n+1, 2n\}$ set into a moduli set with relatively prime moduli, i.e., $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, when n is even, $n \geq 2$, and $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, when n is odd, $n \geq 3$. We demonstrate that such a conversion can be easily done and doesn't require the computation of any multiplicative inverses. We further simplify the 3-moduli CRT for the specific case of $\{2n+2, 2n+1, 2n\}$ moduli set. Second, we propose another new RNS to decimal converter for the moduli set $\{2n+2, 2n+1, 2n\}$ for any integer $n > 0$. We simplify the CRT in order to obtain a reverse converter that utilizes only mod-n instead of mod-$(2n+2)(2n)$ or mod-$2n$ required by other state of the art equivalent converters. Next, we present a low complexity implementation that does not require the explicit calculation of modulo operations in the conversion process as it is normally the case in the traditional CRT and other state of the art equivalent converters. Third, we provide a general 4-moduli RNS conversion scheme and then present a compact form of multiplicative inverses, valid for 4-moduli set $\{2n+3, 2n+2, 2n+1, 2n\}$. We conclude the chapter by assuming a general $\{m_1, m_2, m_3, ..., m_k\}$ moduli set with the dynamic range $M = \prod_{i=1}^{k} m_i$ and introducing a modified CRT that requires mod-m_k instead of mod-M calculations.

- Chapter 5 proposes a novel reverse converter for the moduli set $\{2n+1, 2n, 2n-1\}$. First, we simplify the CRT to obtain a reverse converter that utilizes mod-$(2n-1)$ operations instead of mod-$(2n)(2n-1)$ and mod-$(2n+1)(2n-1)$ operations required by the converters in [41] and [2], respectively. Next, we present a low complexity implementation that does not require explicit modulo operations in the conversion

process as it is normally the case in the traditional CRT and some other state of the art equivalent converters. On the critical path, our proposal requires one 3:1 adder, two 2:1 adders, one multiplier, one comparator, and one multiplexer; the one in [41] requires one 3:1 adder, two 2:1 adders, and two multipliers while the one in [2] requires one 4:1 adder, two 2:1 adders, and two multipliers. Additionally, due to the fact that our scheme operates on smaller magnitude operands, it embeds less complex adders and multipliers, which potentially results in even faster and smaller implementations of the adders and multipliers required by our scheme. In order to get more inside in the practical implications of our proposal, we described our proposal and the converters in [41] and [2] in VHDL and implemented them on Xilinx Spartan 3 FPGA. The synthesis results indicate that, on average, the proposed converter is capable of performing the reverse conversion 14% and 15.8% faster with 27% and 21% area decrease, when compared to the reverse converters in [41] and [2], respectively. The proposed converter also consumes, on average, 8% and 18% less power when compared to the converters in [2] and [41], respectively.

- In Chapter 6, two novel memoryless RNS to binary converters for the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$ are proposed. First, we simplify the CRT to obtain a reverse converter that requires mod-$(2^{n+1}-1)$ instead of both of mod-$(2^{n+1}-1)$ and mod-(2^n-1) required by the best state of the art converter. Second, we further simplify the resulting architecture in order to obtain a reverse converter that utilizes only Carry Save Adders and Carry Propagate Adders. We reduce the large modulo $(2^{n+1}-1)(2^n-1)$ adder to a simple modulo $(2^{n+1}-1)$ adder with the consequence of a reduction in the dynamic range from M to M-$(2^n-1)^2$. We demonstrate that the range reduction penalty decreases exponentially as n increases and for values of $n \geq 5$, the reduction penalty is negligible ($\leq$ 0.0078%). Subsequently, we propose an alternative solution that eliminates the dynamic range limitation at the expense of extra area. We implemented our proposals and the converter in [75] on a Xilinx Spartan 3 FPGA. The results indicate that, on average, the speed of the proposed converter is slightly better when compared with the most effective equivalent state of the art converter. However, the proposed limited dynamic range converter achieves about 42% area reduction, while the second one provides only 29.48% area reduction.

- In Chapter 7, we propose two novel high speed memoryless RNS to binary converters for the moduli set $\{2^{2n+1}-1, 2^n, 2^n-1\}$. First, we simplify the traditional CRT and obtain a new converter, which doesn't cover the entire dynamic range, that only requires mod-$(2^{2n+1}-1)$ operations. Further simplifications result in a very simple and efficient hardware structure, composed of only Carry Save Adders with End-Around Carries and Carry Propagate Adders. Additionally, we resolve the domain restriction problem and propose another reverse converter, which is valid for the entire dynamic range. The theoretical analysis indicates that the two proposed reverse converters are faster than the best known equivalent state of the art reverse converter, but one of them requires more hardware resources. This theoretical result holds true in practice as well as synthesis results suggest that, on average, the proposed limited range CRT and the proposed full range CRT, respectively, are capable of performing the reverse conversion 13.40% and 13.20% faster, with extra costs of 3.29% and 22.75% in terms of area, when compared with the the best known state of the art reverse converter. Additionally, the Area Time Square (AT^2) efficiency metric suggests that the proposed limited range CRT and the proposed full range CRT, respectively, are 24% and 3% more efficient than the state of the art converter.

- Chapter 8 proposes three reverse converters (CI, CII, and CIII) for the new moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. Two of the proposed reverse converters, namely CI and CIII are based on the traditional CRT while the third one, namely CII is based on the MRC. The synthesis results suggest that, CI reduces the conversion computation time by about 6% while also reducing the required hardware area by more than 3%, when compared to CIII. On the other hand, CIII improves the conversion speed by about 17% and reduces the hardware resources by more than 4%, when compared with CII. The proposed CIII is about 11% faster and requires about 22% more hardware resources than the converter in [88]. The AT^2 efficiency metric indicates that it is 5% better than the one in [88].

- Chapter 9 provides concluding remarks on the work presented in this thesis. It summarizes the thesis, outlines its main contributions, and suggests directions for future research.

Chapter 2

Linear Mixed Radix Conversion

As stated in the first chapter, the work on Residue Number System (RNS) to binary/decimal conversion is based on Chinese Remainder Theorem (CRT) [1, 7, 8, 56, 57, 107, 125, 126] or on Mixed Radix Conversion (MRC) [70, 139–141]. CRT is desirable because the computation can be parallelized while MRC is by its very nature a sequential process. However many up to date RNS to binary/decimal converters are based on MRC due to the complex and slow modulo-M operation (M being the system dynamic range thus a rather large constant) required by CRT. The main problem with the MRC is that the computations of the Mixed Radix Digits (MRDs) is done in a serial manner and requires a large number of arithmetic operations.

In this chapter, a MRC based technique is proposed. We introduce an RNS to MRC technique, which addresses the computation of MRDs in such a way that enables the MRC parallelization. With k being the number of moduli, this approach results in RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, while state of the art MRCs exhibit asymptotic complexities in the order of $O\left(k^2\right)$s.

The remainder of this chapter is organized as follows. Section 2.1 introduces the necessary background information, Section 2.2 describes the proposed Linear MRC, the performance of the Linear MRC is evaluated in Section 2.3 while the chapter is concluded in Section 2.4.

2.1 Background

Conversion from RNS to decimal is relatively fast using MRC as it does not involve the large modulo-M calculations present in CRT. Suppose that we have a residue number $(x_1, x_2, x_3, ..., x_k)$ for the moduli set $\{m_1, m_2, m_3, ..., m_k\}$ and a set of digits $\{a_1, a_2, a_3, ..., a_k\}$, which are the MRDs, the decimal equivalent of the residues can be computed as follows [111]:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 + ... + a_n m_1 m_2 m_3 ... m_{k-1} \tag{2.1}$$

where the MRDs are given as follows:

$$\begin{aligned}
a_1 &= x_1 \\
a_2 &= \left| (x_2 - a_1) \left| m_1^{-1} \right|_{m_2} \right|_{m_2} \\
a_3 &= \left| \left((x_3 - a_1) \left| m_1^{-1} \right|_{m_2} - a_2 \right) \left| m_2^{-1} \right|_{m_3} \right|_{m_3} \\
&\cdot \\
&\cdot \\
a_k &= \Big| (((x_k - a_1) \left| m_1^{-1} \right|_{m_k} - a_2) \left| m_2^{-1} \right|_{m_k} - ... \\
&\quad -a_{k-1}) \left| m_{k-1}^{-1} \right|_{m_k} \Big|_{m_k}
\end{aligned} \tag{2.2}$$

Given the MRDs $a_i, 0 \leq a_i < m_i$, any positive number in the interval $[0, \Pi_{i=1}^{k} m_i - 1]$ can be uniquely represented. We note inhere that Equation (2.2) is exactly the same as the one also presented in [70] and parallelizing Equation (2.2) has also been suggested in [70].

One can easily deduce from Equation (2.2) and also in line with the discussion in [141] that a total of $\frac{k(k-1)}{2}$ arithmetic subtractions and multiplications are required. This means that the conversion process that computes the MRDs for an RNS with an k-moduli set has an asymptotic complexity in the order of $O(k^2)$, in terms of the number of arithmetic operations. In the general case, due to the fact that all the m_i products in Equation (2.1) can be precalculated, $(k-1)$ multiplications and $(k-1)$ additions are required for the conversion of a MR number to binary/decimal. That part of the calculation cannot be diminished as it stands on the very nature of MR representation.

The improved MRC described in [141] requires a total of $\frac{k(k-1)}{2}$ subtractions just as in [111] but it can reduce the arithmetic multiplications to $(k-2)$ instead of $\frac{k(k-1)}{2}$ required in [111] for computing the MRDs. Computation of the MRDs in a faster way based on look up tables has been described in [59,67,

81,89]. The complexity of the MRC described in [59,67,81,89,111,141] either in terms of the number of arithmetic operations or in terms of the required number of look up tables is in the order of $O(k^2)$. The algorithm in [81] is reported to be better than that in [89] and [59]. The algorithm presented in [81] is rather complex as it requires solving $\frac{k(k-1)}{2}$ linear Diophantine Equations (DEs). The algorithm presented in this thesis follows in a certain way the same assumptions and principles as [81] but without the need to solve any DE.

Our proposal is stemming from the fact that in Equation (2.2) not all the ways in the modulo-m_i functional units are utilized at each iteration. Given the fact that in one RNS operation all the modulo-m_i units can execute useful computation, the conversion algorithm, which directly follows Equation (2.2) is under-utilizing the available hardware in the RNS processor functional units. Based on this observation and in a more simpler manner when compared to [81], we present in the next section a new method to compute the MR digits a_i that exhibits more parallelism and entirely utilizes the modulo-m_i ways in the RNS processor functional units.

2.2 Modification of Knuth/Szabo and Tanaka's MRC

The MRC as described in Equation (2.2) is by its very nature serial. That is a_i depends on a_j, $i = 2, 3, ..., k$, and $j = 1, 2, 3..., i-1$. To improve the conversion performance, we seek ways to relax these dependencies and make the computation of a_i as parallel as possible. One possible way to do this is based on the fact that a functional unit in an RNS with the moduli $\{m_i\}_{i=1,k}$ has k parallel computation ways. However, at each iteration i in Equation (2.2) not all the modulo ways in the functional units can be utilized in parallel due to the very nature of the evaluated expression. Given that those ways are computing independently, one can reduce the number of operations required in the conversion process by proposing an algorithm that targets the complete utilization of the modulo-m_i ways in the RNS processor functional units. Based on this observation, we propose a new MRC method that asymptotically speaking requires a linear amount of RNS arithmetic operations.

Suppose we have a residue number $(x_1, x_2, x_3, ..., x_k)$ for the moduli $\{m_1, m_2, m_3, ..., m_k\}$, then the mixed radix digits a_i and the decimal equivalent can be computed using Equations (2.2) and (2.1). As previously mentioned, the challenge is to obtain the MRDs a_i in a more parallel manner. All the derivations in this section are done under the assumption that $1 < m_1 < m_2 < m_3 < ... < m_k$ holds true.

In Equation (2.1), every term except the last, i.e., a_1 is a multiple of m_1. If we take the modulo of both sides of Equation (2.1) with respect to m_1, we obtain:

$$|X|_{m_1} = a_1 \tag{2.3}$$

meaning that $a_1 = x_1$.

In a similar manner, if we subtract a_1 from both sides of Equation (2.1), divide both sides by m_1, and then compute the modulo of both sides with respect to m_2 we obtain:

$$a_2 = \left| \frac{X - a_1}{m_1} \right|_{m_2}, \tag{2.4}$$

which is equivalent to:

$$a_2 = \left| \left| (m_1)^{-1} \right|_{m_2} |(x_2 - x_1)|_{m_2} \right|_{m_2}. \tag{2.5}$$

If we follow the same procedures, we can obtain the values of $(a_3, a_4, ..., a_k)$.

In this thesis, we divide the stages involved in the computation of the MRDs a_i into levels in such a way that we maximize the utilization of the RNS processor functional units. The main idea is to perform at the current level i, apart of the computations required for the calculation of a_i also computations that simplify the calculations in the level $(i + 1)$. For that purpose, we introduce auxiliary variables y_i^j, where the exponent j, $j = 0, 1, 2, ..., k-1$, of y denotes different levels where MRDs are computed and i increases by 1 as we progress from one level to another. For example, for level 0, $y_1^0 = a_1$, for level 1, $y_2^1 = a_2$, etc. Based on that, the MRC we propose can be described as follows:

Level 0: The first level is regarded as level 0 and in this level no calculations are required as indicated by Equation (3.3). This implies that $(k-1)$ levels are required for a system involving k-digit MR conversion.

$$y_1^0 = a_1 \quad \text{and} \quad a_1 = x_1 \tag{2.6}$$

Level 1: In this level, we derive a_2. Clearly, $a_2 = \frac{(X-x_1)}{m_1} < m_2 m_3 ... m_k$. Based on that we can compute:

$$y_{j+1}^1 = \left| \frac{x_{j+1} - x_1}{m_1} \right|_{m_{j+1}}, \tag{2.7}$$

which implies that:

$$y_{j+1}^1 = \left| \left| m_1^{-1} \right|_{m_{j+1}} |(x_{j+1} - x_1)|_{m_{j+1}} \right|_{m_{j+1}}, \tag{2.8}$$

where $j = 1, 2, 3, ..., k-1$. If we put $j = 1$ in the above equation, we obtain a_2. The rest of the values corresponding to $j = 2, 3, ..., k-1$ will be utilized in the next level.

$$j = 1, a_2 = y_2^1 = \left| \frac{x_2 - x_1}{m_1} \right|_{m_2}, \tag{2.9}$$

which implies that:

$$\begin{aligned} y_2^1 &= \left| \left| m_1^{-1} \right|_{m_2} |(x_2 - x_1)|_{m_2} \right|_{m_2} && (2.10) \\ j = 2, y_3^1 &= \left| \left| m_1^{-1} \right|_{m_3} |(x_3 - x_1)|_{m_3} \right|_{m_3} && (2.11) \\ j = 3, y_4^1 &= \left| \left| m_1^{-1} \right|_{m_4} |(x_4 - x_1)|_{m_4} \right|_{m_4} && (2.12) \\ &\ldots \end{aligned}$$

One can easily observe that y_2^1, y_3^1, y_4^1,...,y_n^1 can be computed in parallel by different modulo-m_i ways in the RNS processor functional unit.

Level 2: The main goal of Level 2 is to compute a_3. In a similar manner to Level 1, we have:

$$y_{j+2}^2 = \left| \left| m_2^{-1} \right|_{m_{j+2}} \left| \left(y_{j+2}^1 - y_2^1 \right) \right|_{m_{j+2}} \right|_{m_{j+2}}. \tag{2.13}$$

When $j = 1$, we have:

$$a_3 = y_3^2 = \left| \left| m_2^{-1} \right|_{m_3} \left| \left(y_3^1 - y_2^1 \right) \right|_{m_3} \right|_{m_3}. \tag{2.14}$$

When $j = 2$, we have:

$$y_4^2 = \left| \left| m_2^{-1} \right|_{m_4} \left| \left(y_4^1 - y_2^1 \right) \right|_{m_4} \right|_{m_4}. \tag{2.15}$$

When $j = 3$, we have:

$$y_5^2 = \left| \left| m_2^{-1} \right|_{m_5} \left| \left(y_5^1 - y_2^1 \right) \right|_{m_5} \right|_{m_5}. \tag{2.16}$$

Again one can observe that y_3^2 , y_4^2, y_5^2,...,y_n^2 can be computed in parallel also by different modulo-m_i ways in the RNS processor functional unit.

Level 3: Here, y_{j+3}^3 is given as:

$$y_{j+3}^3 = \left| \left| m_3^{-1} \right|_{m_{j+3}} \left| \left(y_{j+3}^2 - y_3^2 \right) \right|_{m_{j+3}} \right|_{m_{j+3}}. \tag{2.17}$$

When $j = 1$,

$$a_4 = y_4^3 = \left| \left| m_3^{-1} \right|_{m_4} \left| \left(y_4^2 - y_3^2 \right) \right|_{m_4} \right|_{m_4}. \tag{2.18}$$

When $j = 2$,

$$y_5^3 = \left| \left| m_3^{-1} \right|_{m_5} \left| \left(y_5^2 - y_3^2 \right) \right|_{m_5} \right|_{m_5}. \tag{2.19}$$

When $j = 3$,

$$y_6^3 = \left| \left| m_3^{-1} \right|_{m_6} \left| \left(y_6^2 - y_3^2 \right) \right|_{m_6} \right|_{m_6}. \tag{2.20}$$

y_4^3, y_5^3, y_6^3,...,y_n^3 will be used in the next level and can be computed in parallel. The rest of the MRDs are computed in a similar manner and the iteration continues until a_n is computed as $y_{j+(k-1)}^{k-1}$ and

$$a_k = \left| \left| m_{k-1}^{-1} \right|_{m_{j+(k-1)}} \left| \left(y_{j+(k-1)}^{k-2} - y_{k-1}^{k-2} \right) \right|_{m_{j+(k-1)}} \right|_{m_{j+(k-1)}},$$

where $j = 1, 2, 3, ..., k - l$, (l is the level number i.e., $l = 1, 2, ..., k - 1$).

2.3 Performance Analysis

When analyzing the proposed method, one can observe the following:

- No computation is required at Level 0;
- one computation is required in the last level, i.e, the computation of a_k;
- each of the remaining $(n-2)$ levels requires $((k-l), l = 1, 2, ..., k-1)$ computations, where by computation we mean one modulo addition and one modulo multiplication.

One can observe however that the $(k-l)$ per level computations can be done in parallel as they utilize different modulo-m_i ways in the RNS processor functional units. Given that they are equivalent to one computation in the RNS hardware. This implies that the total number of computations that are required in all the levels sums up to $(k - 1)$. Hence, the asymptotic complexity of the proposed technique is in the order of $O(k)$. This constitutes a substantial improvement over the state of the art as the MRCs described in [59, 67, 81, 89], either in terms of the number of arithmetic operations or in terms of the required number of look-up tables, have asymptotic complexities in the order of $O(k^2)$.

Mod	RI [in %]	RII [in %]
3	20	0
4	33.33	5.26
5	42.86	14.29
6	50	21.05
7	55.56	26.53
8	60	31.15
9	63.64	35.14
10	66.67	38.64

Table 2.1: Arithmetic Operations Reduction in %

In order to get a better estimate of the impact of the proposed Linear MRC method in practice, we compute the number of required operations for the classic MRC, the method in [141] (MRC [141]), and the proposed method (IMRC), for moduli sets of lengths 3 to 10. The total number of arithmetic operations is computed based on the assumption that an addition takes one cycle and a multiplication two cycles, thus we consider that one multiplication is equivalent delay wise with two additions. MRC and MRC [141] require $\frac{k(k-1)}{2}$ and $(k-2)$ multiplications, respectively, and the same $\frac{k(k-1)}{2}$ additions for the computation of the MRDs. Additionally, all the methods require $(k-1)$ additions and $(k-1)$ multiplications to compute the decimal number according to Equation (2.1). As the moduli set cardinality increases, the number of arithmetic operations in the traditional MRC grows quadratically while for IMRC and the MRC [141], it increases with a constant factor of 6 and $((8+p), p = 0, 1, 2, ...)$ arithmetic operations, respectively. Thus, MRC [141] also increases quadratically.

The results of this comparison are depicted in Table 2.1, Figure 2.1, and Figure 2.2. Table 2.1 presents the percentage reduction of the total number of arithmetic operations required by IMRC when compared to MRC and MRC [141]. We note here that in Table 2.1, the following notations are utilized: Mod - stands for the number of moduli in the considered RNS; RI - stands for reduction of the total number of arithmetic operations in percentage achieved by IMRC over classical MRC; while RII - stands for reduction of the total number of arithmetic operations in percentage achieved by IMRC over MRC [141]. The percentage reductions achieved by the proposed IMRC is depicted by Figure 2.2. One can observe that IMRC achieves 33.33% and 5.26% reductions with moduli sets of length four when compared with the classic MRC and MRC [141], respectively. For moduli sets of length ten,

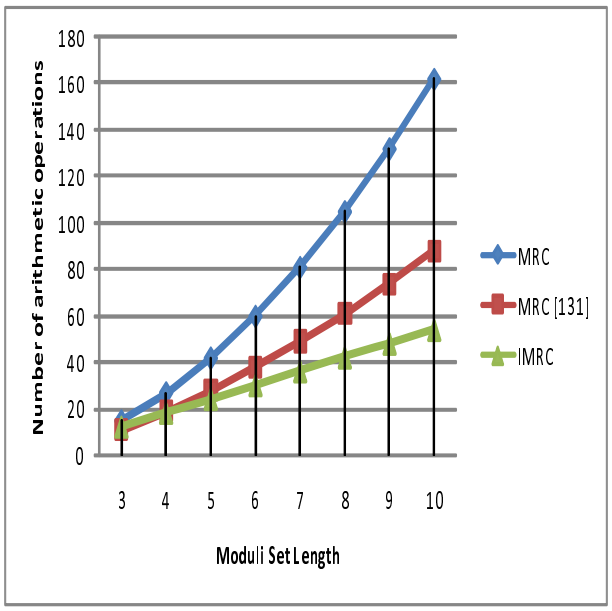


Figure 2.1: Number of arithmetic operations Vs Moduli Set Length

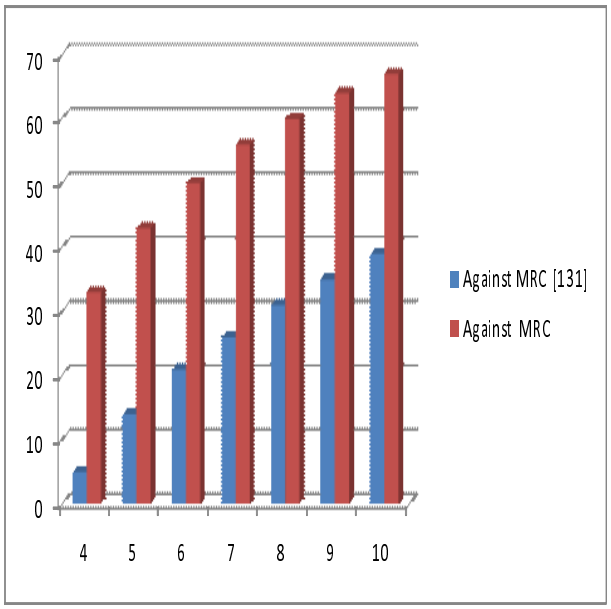


Figure 2.2: % Reduction Vs Moduli Set Length

IMRC achieves 66.67% and 38.64% reductions when compared with the classic MRC and MRC [141], respectively. As expected, the larger the number of moduli in the RNS, the larger the reduction the proposed conversion method exhibits.

The traditional MRC and the one in [141] respectively require $\frac{k(k-1)}{2}$ and $(k-2)$ multiplications with the same $\frac{k(k-1)}{2}$ additions for the computation of MRDs in addition to $(k-1)$ additions and $(k-1)$ multiplications required by Equation (2.1) to compute the decimal number. However, it should be noted that it is not in every case that the improved MRC in [141] reduces the arithmetic multiplications to $(k-2)$. We thus compare the proposed IMRC with this maximum arithmetic reduction in multiplication that can be provided by [141].

Figure 2.1 depicts the IMRC, MRC [141], and the classic MRC performance in terms of the number of arithmetic operations as the length of the moduli set increases. Clearly, it can be seen from Figure 2.1 that the proposed IMRC has the least growth in the RNS arithmetic operations as moduli set length increases when compared with MRC and MRC [141].

2.4 Conclusion

In this chapter, we introduced an RNS to MRC technique, which addresses the computation of MR digits in such a way that enables the MRC parallelization. Given an RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$, the key idea behind the proposed Linear MRC technique is to maximize the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. For an k-digit RNS number $X = (x_1, x_2, x_3, ..., x_k)$ the method requires k iterations. However, at iteration i, the modulo-m_i units are utilized for the calculation of the MR digit a_i, while the other modulo units are calculating intermediate results required in further iterations. Given that one such iteration can be seen as one single operation in the RNS processor functional units, our approach results in an RNS to MR conversion with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, while the traditional MRC and many other state of the art MRCs based techniques exhibit an asymptotic complexity in the order of $O\left(k^2\right)$. More in particular, the utilization of our technique achieved 33.33% and 5.26% reductions with moduli sets of length four when compared with MRC and the improved MRC in [141], respectively. For moduli sets of length ten, our proposal achieved 66.67% and 38.64% reductions when compared with MRC and the improved MRC in [141], respectively.

In the next chapter, we will generalize a previously proposed technique that was restricted to 5-moduli set such that it becomes applicable to any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$. It will be shown that lesser number of arithmetic operations can be obtained using the matrix method.

Chapter 3

Generalized Matrix Method

Given that the Mixed Radix Conversion (MRC) based scheme proposed in Chapter 2 can be improved with respect to the number of arithmetic operations, we present a matrix based method (MATR) in this line of reasoning, which is a generalization of a previously proposed technique that was restricted to 5-moduli set such that it becomes applicable to any Residue Number System (RNS) with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$, where k is the number of moduli. We simplify the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. This scheme results in RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$.

The remainder of this chapter is organized as follows. Section 3.1 introduces the necessary background information, Section 3.2 describes the proposed MATR, Section 3.3 evaluates the MATR's performance, while the chapter is concluded in Section 3.4.

3.1 Background

As said in Chapter 2, performing reverse conversion is very fast using MRC since it does not involve the large modulo-M calculations, which is required by the Chinese Remainder Theorem (CRT). The matrix method described in [140] for the moduli set $\{11, 7, 5, 3, 2\}$, is MRC based. It operates on a number of jumps $J_i, i = 1, 5$ and it was introduced in [140] only by means of an example, the RNS number $(4, 0, 1, 2, 1)$, as follows:

$J_1 = 4$, the first residue and the first location is given by:

$$X - 4 = \begin{pmatrix} |4-4|_{11} = 0 \\ |0-4|_7 = 3 \\ |1-4|_5 = 2 \\ |2-4|_3 = 1 \\ |1-4|_2 = 1 \end{pmatrix}$$

The second jump is defined by a number J_2 such that:

$J_2 = k_2.11$ and $|3 - J_2|_7 = 0$, giving $k_2 = 6$.

The second location is therefore given by:

$$X - 4 - 66 = \begin{pmatrix} |0-66|_{11} = 0 \\ |3-66|_7 = 0 \\ |2-66|_5 = 1 \\ |1-66|_3 = 1 \\ |1-66|_2 = 1 \end{pmatrix}$$

The procedure is repeated until all the elements in the final location are zeros, i.e.,

$$X - 4 - 66 - 231 - 385 - 1155 = \begin{pmatrix} |0|_{11} = 0 \\ |0|_7 = 0 \\ |0|_5 = 0 \\ |0|_3 = 0 \\ |0|_2 = 0 \end{pmatrix}$$

From this description, one can observe that four modular equations must be solved in order to obtain J_is. The required decimal number is given by summing up the J_is. In the next section we present the proposed MATR, which is applicable to any moduli set. Furthermore, we simplify the process of finding the jumps J_i and we show that this process is similar in every stage.

3.2 Proposed Matrix Method

Our approach is based on the periodicity property inherent in RNS. Periodicity is the ability of numbers to cycle in fixed periods with respect to some given moduli and within the dynamic range of the system. For example, given a 3-moduli set $\{m_1 = 5, m_2 = 4, m_3 = 3\}$, the residues cycle in a basic period. That is, the residue sequence of modulus m_1 will have a period of five entries (0,1,2,3,4), m_2 will have a period of four entries (0,1,2,3) while m_3 will have a period of three entries (0,1,2). Based on that, the decimal equiv-

alent of any given residue number is obtained by jumping backwards in the residue table (a table containing all the possible decimal numbers together with their residue equivalent within the dynamic range of the system) to the nearest residue number with at least one residue being zero. The value of that jump is recorded and the process continues until all the residues become zeros. Suppose that we have the residue number $(x_1, x_2, x_3, ..., x_k)$ for the moduli set $\{m_1, m_2, m_3, ..., m_n\}$, we can calculate its decimal counterpart by a maximum number of k-consecutive jumps in the residue table such that each jump increases the number of zero residues with one. The process continues until in the final location all the elements become zero. More formally, given the moduli set $\{m_1, m_2, m_3, ..., m_k\}$, the residue number $(x_1, x_2, x_3, ..., x_k)$ is converted into the decimal number X as follows:

$$X = \sum_{i=1}^{k} p_i, \tag{3.1}$$

where p_i is defined as

$$p_i = (m_1 m_2 ... m_{i-1}) \left| (m_1 m_2 ... m_{i-1})^{-1} \right|_{m_i} \left| t_{(i-1)j} \right|_{m_i}, \tag{3.2}$$

$i > 1$ and $t_{(i-1)j}$ is a value to be determined based on the the following matrix based computation. If the jumps are $(p_1, p_2, p_3, ..., p_k)$ and they correspond to the residues $(x_1, x_2, x_3, ..., x_k)$, respectively, then, $p_1 = x_1$ and the first location is:

$$X - p_1 = \begin{pmatrix} |x_1 - p_1|_{m_1} = 0 \\ |x_2 - p_1|_{m_2} = t_1 \\ |x_3 - p_1|_{m_3} = t_2 \\ |x_4 - p_1|_{m_4} = t_3 \\ |x_5 - p_1|_{m_5} = t_4 \\ . \\ . \\ |x_k - p_1|_{m_k} = t_{k-1} \end{pmatrix} \tag{3.3}$$

The second jump p_2 is given by $p_2 = c_2 m_1$, where c_2 has to satisfy $|t_1 - p_2|_{m_2} = 0$. Solving the above two equations we obtain that

$$p_2 = m_1 \left| \left| (m_1)^{-1} \right|_{m_2} t_1 \right|_{m_2} \tag{3.4}$$

and the second location is defined by:

$$X - p_1 - p_2 = \begin{pmatrix} |0 - p_2|_{m_1} = 0 \\ |t_1 - p_2|_{m_2} = 0 \\ |t_2 - p_2|_{m_3} = t_{21} \\ |t_3 - p_2|_{m_4} = t_{31} \\ |t_4 - p_2|_{m_5} = t_{41} \\ . \\ . \\ |t_k - p_2|_{m_k} = t_{(k-1)1} \end{pmatrix} \tag{3.5}$$

The third jump is determined as $p_3 = c_3 m_1 m_2$ and $|t_{21} - p_3|_{m_3} = 0$ and it is given by:

$$p_3 = (m_1 m_2) \left| \left| (m_1 m_2)^{-1} \right|_{m_3} t_{21} \right|_{m_3} \tag{3.6}$$

The third location is therefore given by:

$$X - p_1 - p_2 - p_3 = \begin{pmatrix} |0 - p_3|_{m_1} = 0 \\ |0 - p_3|_{m_2} = 0 \\ |t_{21} - p_3|_{m_3} = 0 \\ |t_{31} - p_3|_{m_4} = t_{311} \\ |t_{41} - p_3|_{m_5} = t_{411} \\ . \\ . \\ \left|t_{(k-1)1} - p_3\right|_{m_k} = t_{(k-1)11} \end{pmatrix} \tag{3.7}$$

This iterative process continues until the final location is zero and the last jump is determined by $p_k = c_k m_1 m_2 m_3 ... m_{k-1}$ and $\left|t_{(k-1)11...} - p_k\right|_{m_k} = 0$. Solving the above, p_k, which is needed in the last matrix computation is given by:

$$p_k = (m_1 m_2 ... m_{k-1}) \left| \left| (m_1 m_2 ... m_{k-1})^{-1} \right|_{m_k} t_{(k-1)11} \right|_{m_k} \tag{3.8}$$

Then the final location is given by:

$$X - p_1 - p_2 - p_3 ... - p_k = \begin{pmatrix} |0 - p_k|_{m_1} = 0 \\ |0 - p_k|_{m_2} = 0 \\ |0 - p_k|_{m_3} = 0 \\ |0 - p_k|_{m_4} = 0 \\ |0 - p_k|_{m_5} = 0 \\ . \\ . \\ \left|t_{(k-1)11...} - p_k\right|_{m_k} = 0 \end{pmatrix} \tag{3.9}$$

Therefore, as stated in Equation (3.1), the required decimal equivalent of $(x_1, x_2, x_3, ..., x_k)$ with respect to the moduli set $\{m_1, m_2, m_3, ..., m_k\}$ is given by:

$$X = p_1 + p_2 + p_3 + ... + p_k. \tag{3.10}$$

A critical look at Equations (3.3), (3.6), and (3.8) indicates that this process is similar, i.e., it is simply the product of moduli and their multiplicative inverses, which is precomputed together with a number $t_i, i = 1, k - 1$. This is similar to the process of computing MRDs but the subtractions are done in parallel.

In order to clarify the algorithm, let us assume for example that we want to convert $(3, 2, 0)_{RNS(5|4|3)}$ to decimal. The algorithm is applied as follows:

$p_1 = 3$

$$X - 3 = \begin{pmatrix} |3 - 3|_5 = 0 \\ |2 - 3|_4 = 3 \\ |0 - 3|_3 = 0 \end{pmatrix}$$

p_2 is computed by:

$p_2 = 5\,|3|_4 = 15$

The next location is therefore given by:

$$X - 3 - 15 = \begin{pmatrix} |0 - 15|_5 = 0 \\ |3 - 15|_4 = 0 \\ |0 - 15|_3 = 0 \end{pmatrix}$$

The final location is already (0,0,0), there is no need to proceed further and hence the result is $X = 3 + 15 = 18$, as it should.

3.3 Performance Analysis

In the proposed MATR presented in Section 3.2, computation of p_1 is straight forward as $p_1 = x_1$. This implies that the computation of at most $(k-1)p_i$s is required in the conversion process, as in some cases as in the previous example the conversion may need less steps. The computation of each of the $p_i, i = 2, k$ requires 1 multiplication since each $p_i, i = 2, k$ is simply the product of the moduli and their multiplicative invereses, which are pre-computed together with a number $t_i, i = 1, k-1$. This can be clearly seen from Equations (3.3), (3.6), and (3.8). In addition, the conversion process also requires n-parallel subtractions. Given that each of those subtractions is done modulo-m_i, they can be executed in parallel on the RNS adder embedded in the RNS processor. The summation of p_i's also requires $(k-1)$ additions. This implies that the total number of arithmetic operations that are required sums up to at most $(4k-3)$. Hence, the asymptotic complexity of the proposed technique, in terms of the number of arithmetic operation, is in the order of $O(k)$. The total number of operations is computed based on the assumption that an addition takes one cycle and a multiplication two cycles, thus we consider that one multiplication is equivalent delay wise with two additions.

As discussed in Chapter 2, for the MRC, $\frac{k(k-1)}{2}$ additions and $\frac{k(k-1)}{2}$ multiplications are required for the computation of MRDs in addition to $(k-1)$ additions and $(k-1)$ multiplications required in order to compute the desired decimal number while for the proposed MATR, k subtractions, $(k-1)$ multiplications are required in addition to the $(k-1)$ additions required by Equation (3.10) to compute the desired decimal number.

Figure 3.1 shows the behaviour of both the MRC and MATR in terms of the number of arithmetic operations as the length of the moduli set increases. As the moduli set cardinality increases the number of arithmetic operation in MRC grows quadratically while for MATR it increases with a constant factor.

In Table 3.1, we present the percentage reduction of the total number of arithmetic operations required by MATR when compared to MRC. One can observe that MATR achieves 40.00% and 77.16% reductions with moduli set of length three and ten, respectively. As expected, the larger the number of moduli in the RNS, the larger the reduction the proposed conversion method exhibits.

We note here that in Table 3.1, the following notations are utilized: Moduli - stands for the number of moduli in the considered RNS; Reduction - stands for reduction of the total number of arithmetic operations in percentage achieved by MATR over the traditional MRC.

Moduli	Reduction [in %]
3	40.00
4	51.85
5	59.52
6	65.00
7	69.14
8	72.38
9	75.00
10	77.16

Table 3.1: Arithmetic Operations Reduction in %

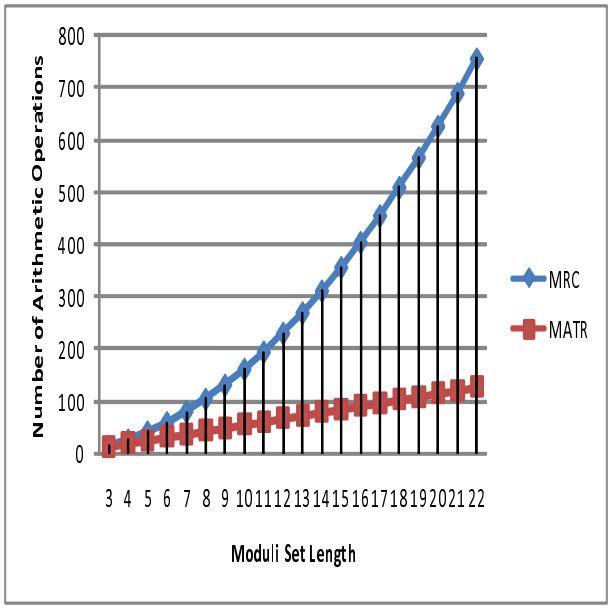


Figure 3.1: Number of Arithmetic Operation vs Moduli Set Length

3.4 Conclusion

In this chapter, we generalized a previously proposed technique that was restricted to 5-moduli set such that it can be utilized in conjunction with any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$. Next, we simplified the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. For a k-digit RNS number $X = (x_1, x_2, x_3, ..., x_k)$ the proposed method takes at most k iterations. Each iteration, except the first one, requires two multiplications and one parallel subtraction over all the mod-m_i ways of the RNS adder. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(k^2\right)$. In particular, the utilization of our technique, for 3-moduli and 10-moduli RNS results in the reduction of the total number of arithmetic operations required by the conversion process with 40.00% and 77.16%, respectively, when compared to the MRC in [139]. Given that the method we proposed substantially reduces the RNS to binary/decimal conversion overhead, it potentially makes RNS more effective in addition and multiplication dominated Digital Signal Processing applications.

Suppose that the large modulo-M operation present in the CRT can be reduced by selecting appropriate moduli sets, the inherent parallelism feature of the CRT will offer greater conversion speed. This is the subject of the next chapter.

Chapter 4

Moduli Sets With Common Factors

The Residue Number System (RNS) for a three moduli set has been studied for a long time with $\{2^n + 1, 2^n, 2^n - 1\}$ being the most popular one [17, 133]. However, the moduli set $\{2n + 2, 2n + 1, 2n\}$ is also considered useful because the numbers are consecutive, enabling nearly equal width adders and multipliers in the hardware implementation [2, 49, 50, 99, 100]. However, the dynamic range provided by the three moduli sets are insufficient in supporting high performance Digital Signal Processing applications requiring a large dynamic range and increased parallelism [121]. Due to this reason, we extend the moduli set $\{2n + 2, 2n + 1, 2n\}$ to the $\{2n + 3, 2n + 2, 2n + 1, 2n\}$ moduli set. Consequently, this chapter proposes efficient reverse converters for the moduli sets $\{2n + 2, 2n + 1, 2n\}$ and $\{2n + 3, 2n + 2, 2n + 1, 2n\}$.

This chapter is organized as follows. Section 4.1 presents the necessary background for the moduli set sharing a common factor. Section 4.2 describes the proposed algorithm for the moduli set $\{2n + 2, 2n + 1, 2n\}$. The second proposed algorithm for the $\{2n + 2, 2n + 1, 2n\}$ moduli set is presented in Section 4.3. Section 4.4 presents the conversion algorithm for the moduli set $\{2n + 3, 2n + 2, 2n + 1, 2n\}$. In Section 4.5, we present a modified CRT, which is applicable to k-moduli set and requires mod-m_k, the smallest modulus instead of the large mod-M operation. Section 4.6 concludes this chapter.

4.1 Background

For a moduli set $\{m_i\}_{i=1,k}$ with the dynamic range $M = \prod_{i=1}^{k} m_i$, the residue number $(x_1, x_2, x_3, ..., x_k)$ can be converted into the decimal number X, according to the Chinese Reminder Theorem (CRT), as follows [111]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M , \tag{4.1}$$

where $M = \prod_{i=1}^{t} m_i$, $M_i = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i. We note here that the moduli set $\{m_i\}_{i=1,t}$ must be pairwise relatively prime for Equation (4.1) to be directly used. The moduli sets $\{2n+2, 2n+1, 2n\}$ and $\{2n+3, 2n+2, 2n+1, 2n\}$ share a common factor. This implies that to utilize Equation (4.1) in the conversion these moduli sets must be first mapped to set of relatively prime moduli. If a moduli set is not pairwise relatively prime, then not every residue set $(x_1, x_2, x_3, ..., x_k)$ corresponds to a number and this results into inconsistency. As discussed in [111], a set of residues is consistent if and only if $|x_i|_t = |x_j|_t$, where $k = gcd(m_i, m_j)$ for all i and j. If this holds true the decimal equivalent of the residue number $(x_1, x_2, x_3, ..., x_k)$ for moduli set which are not pairwise relatively prime can be computed as follows:

$$|X|_{M_L} = \left| \sum_{i=1}^{k} \alpha_i x_i \right|_{M_L} , \tag{4.2}$$

where M_L is the Lowest Common Multiple (LCM) of $\{m_i\}_{i=1,k}$, the set of moduli sharing a common factor, X is the decimal equivalent of $\{x_i\}_{i=1,k}$, α_i is an integer such that $|\alpha_i|_{\frac{M_L}{\mu_i}} = 0$ and $|\alpha_i|_{\mu_i} = 1$, and $\{\mu_i\}_{i=1,k}$ is a set of integers such that $M_L = \prod_{i=1}^{k} \mu_i$ and μ_i divides m_i. It should be noted that α_i may not exist for some i.

4.2 Modulo-m_3 Conversion Technique

The main idea behind our approach is to simplify Equation (4.1) by eliminating the large modulo M and by removing the cost of computing M_i^{-1}. In this section, we demonstrate that the first is possible for any 3-moduli RNS, while

the second one can be achieved only for 3-moduli sets which are not pairwise relatively prime. We first introduce a modified CRT for general moduli set of length three, which does not require mod-M computations.

Theorem 4.1 *For a moduli set $\{m_i\}_{i=1,3}$ the decimal equivalent X of the residue number (x_1, x_2, x_3) can be computed as:*

$$X = (x_1 + x_2) + m_1 m_2 \left| k_1 x_1 + k_2 x_2 + \left| M_3^{-1} \right|_{m_3} x_3 \right|_{m_3}, \tag{4.3}$$

where M_3^{-1} is the multiplicative inverse of M_3, $k_1 = \frac{(M_1 \left| M_1^{-1} \right|_{m_1} - 1)}{m_1 m_2}$ and $k_2 = \frac{(M_2 \left| M_2^{-1} \right|_{m_2} - 1)}{m_1 m_2}$.

Proof:
We use lemmas presented earlier in [130]:

Lemma 1: $|am_1|_{m_1 m_2} = m_1 |a|_{m_2}$

Lemma 2: $M_1 \left| M_1^{-1} \right|_{m_1} = 1 + k_1 m_1 m_2$

Lemma 3: $M_2 \left| M_2^{-1} \right|_{m_2} = 1 + k_2 m_1 m_2$

Expanding Equation (4.1) for $k = 3$ we obtain:

$$X = \left| M_1 \left| M_1^{-1} \right|_{m_1} x_1 + M_2 \left| M_2^{-1} \right|_{m_2} x_2 + M_3 \left| M_3^{-1} \right|_{m_3} x_3 \right|_{m_1 m_2 m_3} \tag{4.4}$$

Using Lemma 2 and 3 in the above equation, we have:

$$X = |(1 + k_1 m_1 m_2) x_1 + (1 + k_2 m_1 m_2) x_2 + M_3 \left| M_3^{-1} \right|_{m_3} x_3|_{m_1 m_2 m_3} \tag{4.5}$$

Further simplification gives:

$$X = (x_1 + x_2) + |k_1 m_1 m_2 x_1 + k_2 m_1 m_2 x_2 + M_3 \left| M_3^{-1} \right|_{m_3} x_3|_{m_1 m_2 m_3} \tag{4.6}$$

Applying Lemma 1, we get:

$$X = (x_1 + x_2) + m_1 m_2 |k_1 x_1 + k_2 x_2 + M_3^* \left| M_3^{-1} \right|_{m_3} x_3|_{m_3} \tag{4.7}$$

Here, $M_3^* = \frac{M_3}{m_1 m_2} = 1$, the above equation then reduces to:

$$X = (x_1 + x_2) + m_1 m_2 | k_1 x_1 + k_2 x_2 + \left| M_3^{-1} \right|_{m_3} x_3 |_{m_3} \tag{4.8}$$

It can be observed that Equation (4.8) makes use of mod-m_3 (the smallest modulus) instead of mod-M operations, thus the magnitude of involved values is smaller than in the traditional CRT, and that k_1 and k_2 can be precomputed.

The next simplification step is the elimination of the M_i^{-1}. To achieve that, we restrict to $\{2n+2, 2n+1, 2n\}$ moduli set and first introduce a formal representation of Equation (4.2).

Theorem 4.2 *For a moduli set $\{m_i\}_{i=1,k}$ sharing a common factor, which must first be mapped into a set of pairwise relatively prime moduli, $\{\mu_i\}_{i=1,k}$, the decimal equivalent X of the residue number $(x_1, x_2, x_3, ..., x_k)$ can be computed as:*

$$|X|_{M_L} = \left| \sum_{i=1}^{k} \beta_i \left| \beta_i^{-1} \right|_{\mu_i} x_i \right|_{M_L}, \tag{4.9}$$

where $M_L = LCM\{m_i\}_{i=1}^n = \prod_{i=1}^{n} \mu_i$, $\beta_i = \frac{M_L}{\mu_i}$, $\left| \beta_i^{-1} \right|_{\mu_i}$ is the multiplicative inverse of β_i with respect to μ_i.

Proof:
To demonstrate the correctness of Equation (4.9), we shall relate it to Equation (4.2). All the conditions in Equation (4.2) have been taken care of in our mapping formula except for the condition that α_i is an integer such that $|\alpha_i|_{\frac{M_L}{\mu_i}} = 0$ and $|\alpha_i|_{\mu_i} = 1$. Let us assume that $\alpha_i = \beta_i * p$. This implies that $|\beta_i * p|_{\mu_i} = 1$, meaning that: $p = \left| \beta_i^{-1} \right|_{\mu_i}$. We can then write $\alpha_i = \beta_i * \left| \beta_i^{-1} \right|_{\mu_i}$ and this is what we have in Equation (4.9). We then show that $|\alpha_i|_{\frac{M_L}{\mu_i}} = 0$. $|\alpha_i|_{\frac{M_L}{\mu_i}} = \left| \beta_i * \left| \beta_i^{-1} \right|_{\mu_i} \right|_{\frac{M_L}{\mu_i}}$, which implies that $|\alpha_i|_{\frac{M_L}{\mu_i}} = \left| \frac{M_L}{\mu_i} * \left| \beta_i^{-1} \right|_{\mu_i} \right|_{\frac{M_L}{\mu_i}}$ since $\beta_i = \frac{M_L}{\mu_i}, |\alpha_i|_{\frac{M_L}{\mu_i}} = 0$. Hence, Equation (4.9) is a formal representation of Equation (4.2).
To utilize Equation (4.9) in the conversion, we need a method to compute the relatively prime $\{\mu_i\}_{i=1,k}$ for a moduli set $\{m_i\}_{i=1,k}$ sharing a common factor. According to [2] and [100] the moduli set $\{2n+2, 2n+1, 2n\}$ with a

common factor can be mapped to a set of pairwise relatively prime moduli, μ_i given by:

1. $\{n+1, 2n+1, 2n\}$, which implies that the new moduli set is $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, when n is even, $n \geq 2$,
2. $\{2n+2, 2n+1, n\}$, which implies that the new moduli set is $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, when n is odd, $n \geq 3$.

The conditions $(n \geq 2)$ and $(n \geq 3)$ are very important as due to them $\mu_i > 1$ meaning that $\left|\beta_i^{-1}\right|_{\mu_i}$ in Equation (4.9) and consequently α_i in Equation (4.2) always exists. Based on this, we carry out a C++ program simulation for the moduli set under investigation in order to obtain the LCM of the moduli set for different values of n and taking the conditions $M_L = \prod_{i=1}^{n} \mu_i$ and $|m_i|_{\mu_i} = 0$ into consideration. From the pattern of results obtained, we could see that mapping from m_i to μ_i can be done by classifying the values of n into even and odd, which is in good agreement with what has been suggested in [2] and [100]. Moreover, based on the collected experimental data, some of which are displayed in Tables 4.3, 4.4, 4.5, and 4.6, we observe that there is a well established relationship between the various multiplicative inverses and the moduli. Deductions are made based on this observation and the relations are presented in Tables 4.1 and 4.2. The experimental results presented in Table 4.1, using the new set $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, suggest that the following relations exist between the moduli and the multiplicative inverses:

$$\left|\mu_1^{-1}\right|_{\mu_3} = \tfrac{m_1}{2}, \left|(\mu_1\mu_2)^{-1}\right|_{\mu_3} = \tfrac{m_1}{2},$$
$$\left|(\mu_2\mu_3)^{-1}\right|_{\mu_1} = \tfrac{m_3}{4} + 1, \left|(\mu_1\mu_3)^{-1}\right|_{\mu_2} = m_2 - 2,$$
$$\left|\mu_1^{-1}\right|_{\mu_2} = 2.$$

Similarly, Table 4.2, using the new set $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, suggests that the following hold true:

$$\left|\mu_1^{-1}\right|_{\mu_3} = \tfrac{m_1}{4}, \left|(\mu_1\mu_2)^{-1}\right|_{\mu_3} = \tfrac{m_1}{4},$$
$$\left|(\mu_2\mu_3)^{-1}\right|_{\mu_1} = m_1 - \tfrac{m_3}{2}, \left|(\mu_1\mu_3)^{-1}\right|_{\mu_2} = m_2 - 2,$$
$$\left|\mu_1^{-1}\right|_{\mu_2} = 2.$$

For moduli sets with a common factor not all remainder sets are valid numbers. The following proposition states the condition for a 3-residue set to represent a valid number.

Proposition 4.1 *For RNS with moduli set* $\{m_1, m_2, m_3\}$ *sharing a common factor,* (x_1, x_2, x_3) *represents a valid number if and only if* $(x_1 + x_3)$ *is even.*

Proof: This proposition has been proved in [2].

Making the appropriate substitutions in Theorem 4.1, we can particularize it for 3-moduli RNS sharing a common factor as follows:

Corollary 4.1 *For the moduli set* $\{2n + 2, 2n + 1, 2n\}$ *the decimal equivalent* X *of the residue number* (x_1, x_2, x_3)*,* $(x_1 + x_3)$ *being even, can be computed as follows:*

1. *If* n *is* ***even****:*

$$X = (x_1 + x_2) + \frac{m_1 m_2}{2} \left| k_1 x_1 + k_2 x_2 + \frac{m_1}{2} x_3 \right|_{m_3}, \tag{4.10}$$

where

$$\begin{aligned} k_1 &= \frac{2((m_2 m_3)(\frac{m_3}{4} + 1) - 1)}{(m_1 m_2)}, \\ k_2 &= \frac{2(\frac{(m_1 m_3)}{2}(m_2 - 2) - 1)}{m_1 m_2}. \end{aligned}$$

2. *If* n *is* ***odd****:*

$$X = (x_1 + x_2) + m_1 m_2 \left| k_1 x_1 + k_2 x_2 + \frac{m_1}{4} x_3 \right|_{\frac{m_3}{2}}, \tag{4.11}$$

where

$$\begin{aligned} k_1 &= \frac{(\frac{m_2 m_3}{2}(m_1 - \frac{m_3}{2}) - 1)}{m_1 m_2}, \\ k_2 &= \frac{(\frac{m_1 m_3}{2}(m_2 - 2) - 1)}{m_1 m_2}. \end{aligned}$$

S/N	Multiplicative Inverses	Equivalent Values
1	$\left\|\mu_1^{-1}\right\|_{\mu_2}$	2
2	$\left\|\mu_2^{-1}\right\|_{\mu_3}$	1
3	$\left\|\mu_1^{-1}\right\|_{\mu_3}$	$\frac{m_1}{2}$
4	$\left\|(\mu_1\mu_2)^{-1}\right\|_{\mu_3}$	$\frac{m_1}{2}$
5	$\left\|(\mu_2\mu_3)^{-1}\right\|_{\mu_1}$	$\frac{m_3}{4}+1$
6	$\left\|(\mu_1\mu_3)^{-1}\right\|_{\mu_2}$	m_2-2

Table 4.1: Multiplicative Inverse Value for n Even $(n \geq 2)$

S/N	Multiplicative Inverses	Equivalent Values
1	$\left\|\mu_1^{-1}\right\|_{\mu_2}$	1
2	$\left\|\mu_2^{-1}\right\|_{\mu_3}$	1
3	$\left\|\mu_1^{-1}\right\|_{\mu_3}$	$\frac{m_1}{4}$
4	$\left\|(\mu_1\mu_2)^{-1}\right\|_{\mu_3}$	$\frac{m_1}{4}$
5	$\left\|(\mu_2\mu_3)^{-1}\right\|_{\mu_1}$	$m_1-\frac{m_3}{2}$
6	$\left\|(\mu_1\mu_3)^{-1}\right\|_{\mu_2}$	(m_2-2)

Table 4.2: Multiplicative Inverse Value for n Odd $(n \geq 3)$

n	Given Set	New Set	$\left\|(\mu_1\mu_2)^{-1}\right\|_{\mu_3}$
2	$\{6,5,4\}$	$\{3,5,4\}$	3
4	$\{10,9,8\}$	$\{5,9,8\}$	5
6	$\{14,13,12\}$	$\{7,13,12\}$	7
8	$\{18,17,16\}$	$\{9,17,16\}$	9
10	$\{22,21,20\}$	$\{11,21,20\}$	11
12	$\{26,25,24\}$	$\{13,25,24\}$	13

Table 4.3: Mapping for n Even and Multiplicative Inverse

n	$\left\|\mu_1^{-1}\right\|_{\mu_3}$	$\left\|(\mu_2\mu_3)^{-1}\right\|_{\mu_1}$	$\left\|(\mu_1\mu_3)^{-1}\right\|_{\mu_2}$
2	3	2	3
4	5	3	7
6	7	4	11
8	9	5	15
10	11	6	19
12	13	7	23

Table 4.4: Multiplicative Inverse Values for n Even

n	Given Set	New Set	$\left\|(\mu_1\mu_2)^{-1}\right\|_{\mu_3}$
3	$\{8, 7, 6\}$	$\{8, 7, 3\}$	2
5	$\{12, 11, 10\}$	$\{12, 11, 5\}$	3
7	$\{16, 15, 14\}$	$\{16, 15, 7\}$	4
9	$\{20, 19, 18\}$	$\{20, 19, 9\}$	5
11	$\{24, 23, 22\}$	$\{24, 23, 11\}$	6
13	$\{28, 27, 26\}$	$\{28, 27, 13\}$	7

Table 4.5: Mapping for n Odd and Multiplicative Inverse

n	$\left\|\mu_1^{-1}\right\|_{\mu_3}$	$\left\|(\mu_2\mu_3)^{-1}\right\|_{\mu_1}$	$\left\|(\mu_1\mu_3)^{-1}\right\|_{\mu_2}$
3	2	5	5
5	3	7	9
7	4	9	13
9	5	11	17
11	6	13	21
13	7	15	25

Table 4.6: Multiplicative Inverse Values for n Odd

Operations	[2]	Proposed Algorithm
Additions	5	4
Multiplications	4	4
Reduced M	m_3m_1	m_3 or $\frac{m_3}{2}$

Table 4.7: Performance Comparison

Proof:
Trivial with proper substitutions from Tables 4.1 and 4.2 and due to Proposition 4.1.

Clearly, it can be seen that the numbers involved in the multiplication are very small when compared to the numbers involved in the direct CRT implementation. Additionally, the large modulo M calculations are replaced by modulo calculations with the smallest modulus in the moduli set under consideration. Previous work on 3-moduli RNS in [2, 100] has demonstrated improvement over traditional CRT in terms of operands magnitude as this determines the complexity and delay of the associated RNS hardware. Additionally, [2] outperformed [100] in terms of the operands magnitude, thus we compare our proposal with this approach. As indicated in Table 4.7, our proposal requires less arithmetic operations and even more important for the hardware, complexity of the operands magnitude is significantly reduced. More specifically, the modulo operation has been reduced from modulo $M = m_1 m_3$ to modulo m_3 or $\frac{m_3}{2}$. In the next section, we present another novel conversion scheme, which does not require the computation of modulo operation.

4.3 Computation without Modulo Operation

Given the RNS number (x_1, x_2, x_3) for the moduli set $\{2n+2, 2n+1, 2n\}$, the proposed algorithm computes the decimal equivalent of this RNS number based on a further simplification of the well-known traditional CRT. First, we show that the computation of the multiplicative inverses can be eliminated for this moduli set. Next, we propose a low complexity implementation that does not require explicit use of the modulo operation at the final stage of computation. As demonstrated in Section 4.2, the set of relatively prime moduli for $\{2n+2, 2n+1, 2n\}$ moduli set for both even and odd integers $n > 0$ are given by $\{n+1, 2n+1, 2n\}$ and $\{2n+2, 2n+1, n\}$, respectively. This implies that the new moduli sets for n-even and-odd are $\left\{\frac{m_1}{2}, m_2, m_3\right\}$ and $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, respectively.
The following theorem shows the existence of the compact forms of multiplicative inverses for any even integer $n > 0$.

Theorem 4.3 *Given the moduli set* $\{2n+2, 2n+1, 2n\}$ *for even integer* $n > 0$

with $m_1 = 2n+2, m_2 = 2n+1, m_3 = 2n$*, the following hold true:*

$$\left|(\frac{m_1}{2}m_2)^{-1}\right|_{m_3} = n+1, \tag{4.12}$$

$$\left|(m_2m_3)^{-1}\right|_{\frac{m_1}{2}} = \frac{n}{2}+1, \tag{4.13}$$

$$\left|(\frac{m_1}{2}m_3)^{-1}\right|_{m_2} = 2n-1. \tag{4.14}$$

Proof:
If it can be demonstrated that $\left|(n+1)\times(\frac{m_1}{2}m_2)\right|_{m_3} = 1$, then $(n+1)$ is the multiplicative inverse of $(\frac{m_1}{2}m_2)$ with respect to m_3. $\left|(n+1)\times(\frac{m_1}{2}m_2)\right|_{m_3}$ is given by: $|(n+1)(n+1)(2n+1)|_{2n} = \left|(2n^3+5n^2+4n+1)\right|_{2n} = |\left|2n(n^2+\frac{5n}{2})\right|_{2n} + |2(2n)|_{2n} + |1|_{2n}|_{2n} = |0+0+1|_{2n} = 1$, thus Equation (4.12) holds true.

In the same way, if $\left|(\frac{n}{2}+1)\times(m_2m_3)\right|_{\frac{m_1}{2}} = 1$, then $(\frac{n}{2}+1)$ is the multiplicative inverse of (m_2m_3) with respect to $\frac{m_1}{2}$. $\left|(\frac{n}{2}+1)\times(m_2m_3)\right|_{\frac{m_1}{2}}$ is given by: $|(\frac{n}{2}+1)(2n+1)(2n)|_{n+1} = |2n^3+5n^2+2n|_{n+1} = ||2n^2(n+1)|_{n+1} + |3n^2+2n|_{n+1}|_{n+1} = |0+1|_{n+1} = 1$, thus Equation(4.13) holds true.

Again, if $|(2n-1)\times(\frac{m_1}{2}m_3)|_{m_2} = 1$, then $2n-1$ is the multiplicative inverse of $(\frac{m_1}{2}m_3)$ with respect to m_2. $|(2n-1)\times(\frac{m_1}{2}m_3)|_{m_2}$ is given by: $|(2n-1)(n+1)(2n)|_{2n+1} = |4n^3+2n^2-2n|_{2n+1} = ||2n^2(2n+1)|_{2n+1} + |-2n|_{2n+1}|_{2n+1} = |0+1|_{2n+1} = 1$, thus Equation (4.14) holds true.

The following theorem introduces a simplified way to compute the decimal equivalent of the RNS number (x_1, x_2, x_3).

Theorem 4.4 *The decimal equivalent of the RNS number* (x_1, x_2, x_3) *for the moduli set* $\{m_1, m_2, m_3\}$ *in the form* $\{2n+2, 2n+1, 2n\}$ *for any even integer* $n > 0$ *is computed as follows:*

$$X = \left|\frac{m_2m_3}{2}x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_{M_L}, \tag{4.15}$$

where $M_L = \frac{m_1m_2m_3}{2}$.

Proof:
For $t = 3$, Equation (4.2) becomes:

$$X = \left|\sum_{i=1}^{3} M_i \left|M_i^{-1}x_i\right|_{m_i}\right|_{M_L}. \tag{4.16}$$

By substituting Equations (4.12), (4.13), and (4.14) into Equation (4.16) we obtain the following:

$$\begin{aligned} X &= \Big|(m_2m_3)\Big(\frac{m_3}{4}+1\Big)x_1+\Big(\frac{m_1}{2}m_3(m_2-2)\Big)x_2 \\ &\quad +\Big(\frac{m_1}{2}m_2\Big)\frac{m_1}{2}x_3\Big|_{M_L} \\ &= \Big|\Big(\frac{m_2m_3m_3}{4}\Big)x_1+m_2m_3x_1+\frac{m_1m_2m_3}{2}x_2 \\ &\quad -m_1m_3x_2+\frac{m_1m_1m_2}{4}x_3\Big|_{M_L} \\ &= \Big|\Big(\frac{m_2m_3}{4}\Big)x_1(m_1-2)+m_2m_3x_1+M_Lx_2 \\ &\quad -m_1m_3x_2+\frac{m_1m_2}{4}x_3(m_3+2)\Big|_{M_L} \end{aligned}$$

Further simplifications give:

$$\begin{aligned} X &= \Big|\Big|\frac{M_L}{2}(x_1+x_3)\Big|_{M_L}+\Big|\Big(\frac{m_2m_3}{2}\Big)x_1\Big|_{M_L} \\ &\quad -|m_1m_3x_2|_{M_L}+\Big|\Big(\frac{m_1m_2}{2}\Big)x_3\Big|_{M_L}\Big|_{M_L} \end{aligned} \tag{4.17}$$

Since each of the terms $\frac{m_2m_3}{2}x_1$, $m_1m_3x_2$, and $\frac{m_1m_2}{2}x_3$ in Equation (4.17) is positive and less than M_L and also from Proposition 4.1, (x_1+x_3) must always be even, which implies that, $\Big|(x_1+x_3)\frac{M_L}{2}\Big|_{M_L}=0$. Equation (4.17) therefore reduces to:

$$X = \Big|\frac{m_2m_3}{2}x_1-m_1m_3x_2+\frac{m_1m_2}{2}x_3\Big|_{M_L} \tag{4.18}$$

Thus, Equation (4.15) holds true.

In the following theorem, we demonstrate that compact forms of multiplicative inverses do exist also for any odd integer $n>0$.

Theorem 4.5 *Given the moduli set* $\{m_1,m_2,m_3\}$ *with* $m_1=2n+2, m_2=$

$2n+1, m_3 = 2n$ *for any odd integer* n*, the following hold true:*

$$\left|(m_1m_2)^{-1}\right|_{\frac{m_3}{2}} = \frac{n+1}{2}, \tag{4.19}$$

$$\left|\left(m_2\frac{m_3}{2}\right)^{-1}\right|_{m_1} = n+2, \tag{4.20}$$

$$\left|\left(m_1\frac{m_3}{2}\right)^{-1}\right|_{m_2} = 2n-1. \tag{4.21}$$

Proof:
If it can be shown that $\left|\frac{(n+1)}{2} \times (m_1m_2)\right|_{\frac{m_3}{2}} = 1$, then $\frac{(n+1)}{2}$ is the multiplicative inverse of (m_1m_2) with respect to $\frac{m_3}{2}$. $\left|\frac{(n+1)}{2} \times (m_1m_2)\right|_{\frac{m_3}{2}}$ is given by: $|(n+1)(n+1)(2n+1)|_n = |(n^2+2n+1)(2n+1)|_n = |2n^3+5n^2+4n+1|_n = ||2n^3|_n+|5n^2|_n+|4n|_n+|1|_n|_n = |0+0+0+1|_n = 1$, thus Equation (4.19) holds true.

In the same way, if $\left|(n+2) \times \left(m_2\frac{m_3}{2}\right)\right|_{m_1} = 1$, then $(n+2)$ is the multiplicative inverse of $\left(m_2\frac{m_3}{2}\right)$ with respect to m_1. $\left|(n+2)\left(m_2\frac{m_3}{2}\right)\right|_{m_1}$ is given by: $|(n+2)(2n+1)(n)|_{2n+2} = |(n+2)(2n^2+n)|_{2n+2} = |2n^3+n^2+4n^2+2n|_{2n+2} = |2n^3+2n^2+n^2+n(2n+2)|_{2n+2} = |(n^2+n)(2n+2)+n^2|_{2n+2} = ||(n^2+n)(2n+2)|_{2n+2}+|n^2|_{2n+2}|_{2n+2} = |0+1|_{2n+2} = 1$, thus Equation (4.20) holds true.

Again, if $|\left((2n-1) \times m_1\frac{m_3}{2}\right)|_{2n+1} = 1$, then $2n-1$ is the multiplicative inverse of $m_1\frac{m_3}{2}$ with respect to m_2. $|\left((2n-1) \times m_1\frac{m_3}{2}\right)|_{2n+1}$ is given by $|(2n-1)(2n+2)(n)|_{2n+1} = |(2n-1)(2n^2+2n)|_{2n+1} = |4n^3+4n^2-2n^2-2n|_{2n+1} = |4n^3+2n^2-2n|_{2n+1} = |2n^2(2n+1)-2n|_{2n+1} = ||2n^2(2n+1)|_{2n+1}+|-2n|_{2n+1}|_{2n+1} = |0+1|_{2n+1} = 1$, thus Equation (4.21) holds true.

The following theorem introduces a simplified way to compute the decimal equivalent of the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+2, 2n+1, 2n\}$ for any odd integer $n > 0$.

Theorem 4.6 *The decimal equivalent of the RNS number* (x_1, x_2, x_3) *for the moduli set* $\{2n+2, 2n+1, 2n\}$ *for any odd integer* $n > 0$ *is given by:*

$$X = \left|\frac{m_2m_3}{2}x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_{M_L}, \tag{4.22}$$

where $M_L = \frac{m_1m_2m_3}{2}$.

Proof:
Given that Equation (4.22) was proved to be true for n-even in Equation (4.15) above, we wish to show also its validity for n-odd.
For $t = 3$, Equation (4.2) becomes:

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_{M_L}. \tag{4.23}$$

By substituting Equations (4.19), (4.20), and (4.21) into Equation (4.23) we obtain the following:

$$\begin{aligned} X &= \left| \left(m_2 \frac{m_3}{2} \right) (m_1 + 2) x_1 + \left(m_1 \frac{m_3}{2} \right) (m_2 - 2) x_2 \right. \\ &\quad \left. + (m_1 m_2) \frac{(m_3 + 2)}{4} \right|_{M_L} \\ &= \left| \frac{m_1 m_2 m_3}{4} x_1 + \frac{m_2 m_3}{2} x_1 + \frac{m_1 m_2 m_3}{2} x_2 \right. \\ &\quad \left. - m_1 m_3 x_2 + \frac{m_1 m_2 m_3}{4} x_3 + \frac{m_1 m_2}{2} x_3 \right|_{M_L} \\ &= \left| \left(\frac{m_2 m_3}{4} \right) x_1 (m_1 - 2) + m_2 m_3 x_1 + M_L x_2 \right. \\ &\quad \left. - m_1 m_3 x_2 + \frac{m_1 m_2}{4} x_3 (m_3 + 2) \right|_{M_L} \end{aligned}$$

Further simplifications give:

$$\begin{aligned} X &= \left| \left| \frac{M_L}{2} (x_1 + x_3) \right|_{M_L} + \left| \left(\frac{m_2 m_3}{2} \right) x_1 \right|_{M_L} \right. \\ &\quad \left. - |m_1 m_3 x_2|_{M_L} + \left| \left(\frac{m_1 m_2}{2} \right) x_3 \right|_{M_L} \right|_{M_L} \end{aligned} \tag{4.24}$$

Since each of the terms $\frac{m_2 m_3}{2} x_1$, $m_1 m_3 x_2$, and $\frac{m_1 m_2}{2} x_3$ in Equation (4.24) is positive and less than M_L and also from Proposition 4.1, $(x_1 + x_3)$ must always be even, which implies that, $\left| (x_1 + x_3) \frac{M_L}{2} \right|_{M_L} = 0$. Equation (4.24) therefore reduces to:

$$X = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_{M_L} \tag{4.25}$$

Thus, Equation (4.22) holds true for both n-even and odd.

We wish to further simplify Equation (4.22) using the following theorem.

Theorem 4.7 *The decimal equivalent of the RNS number (x_1, x_2, x_3) for the moduli set $\{2n+2, 2n+1, 2n\}$ is computed as follows:*

$$\begin{cases} X = y, & y \geq 0 \\ X = y + M, & y<0 \end{cases} \tag{4.26}$$

where

$$\begin{aligned} y &= m_2(x_2 - x_1) + x_1 \\ &\quad + m_1 m_2 \left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \end{aligned} \tag{4.27}$$

Proof:
To prove this theorem we use the following lemma presented in [130]:

$$|a m_1|_{m_1 m_2} = m_1 \, |a|_{m_2} . \tag{4.28}$$

From Equation (4.22), we have

$$X = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_{M_L}$$

Putting $m_3 = m_2 - 1$ in the above equation, we obtain:

$$\begin{aligned} X &= \left| \frac{m_2 m_3}{2} x_1 - m_1 x_2 (m_2 - 1) + \frac{m_1 m_2}{2} x_3 \right|_{M_L} \\ &= |m_1 x_2 \\ &\quad + \left| \frac{m_2 m_3}{2} x_1 - m_1 m_2 x_2 + \frac{m_1 m_2}{2} x_3 \right|_{\frac{m_1 m_2 m_3}{2}} \Bigg|_{M_L} \end{aligned}$$

Applying Equation (4.28) we have:

$$\begin{aligned} X &= |m_1 x_2 \\ &\quad + m_2 \left| \frac{m_3}{2} x_1 - m_1 x_2 + \frac{m_1}{2} x_3 \right|_{\frac{m_1 m_3}{2}} \Bigg|_{M_L} \end{aligned} \tag{4.29}$$

Putting $m_3 = m_1 - 2$ in the above equation, we obtain:

$$\begin{aligned} &= |m_1 x_2 \\ &\quad + m_2 \left| \frac{(m_1 - 2)}{2} x_1 - m_1 x_2 + \frac{m_1}{2} x_3 \right|_{\frac{m_1 m_3}{2}} \Bigg|_{M_L} \\ &= |m_1 x_2 - m_2 x_1 \\ &\quad + m_2 \left| m_1 \frac{(x_1 + x_3)}{2} - m_1 x_2 \right|_{\frac{m_1 m_3}{2}} \Bigg|_{M_L} \end{aligned}$$

Applying Equation (4.28) we obtain:

$$\begin{aligned} X &= |m_1x_2 - x_1(m_1-1) \\ &\quad +m_1m_2\left|\frac{(x_1+x_3)}{2} - x_2\right|_{\frac{m_3}{2}}\Bigg|_{M_L} \end{aligned}$$

Further simplifications give:

$$\begin{aligned} X &= |(x_2-x_1)m_1 + x_1 \\ &\quad +m_1m_2\left|\frac{(x_1+x_3)}{2} - x_2\right|_{\frac{m_3}{2}}\Bigg|_{M_L} \end{aligned} \tag{4.30}$$

Equation (4.30) is the general expression of Equation (4.27), valid for both y positve and negative. The next stage of the proof is to demonstrate that at most one M_L corrective addition is required. By definition of modulus we have:

$$\begin{aligned} 0 \leq \quad & \left|\tfrac{x_1+x_3}{2} - x_2\right|_{\frac{m_3}{2}} && \leq \frac{m_3}{2} - 1 && \Big| \cdot m_1m_2 \\ 0 \leq \quad & m_1m_2\left|\tfrac{x_1+x_3}{2} - x_2\right|_{\frac{m_3}{2}} && \leq \frac{m_1m_2m_3}{2} - m_1m_2. \end{aligned}$$

Adding to this double inequality the following inequalities:
$0 \leq m_1x_2 < m_1m_2$ and $-m_1m_2 < -m_2x_1 \leq 0$,
we have

$$-m_1m_2 < m_1x_2 - m_2x_1 + m_1m_2\left|\frac{x_1+x_3}{2} - x_2\right|_{\frac{m_3}{2}} < \frac{m_1m_2m_3}{2}$$

Thus one corrective addition of M_L is required in order to obtain the correct result when $y < 0$, and (4.27) holds true.

For the case when $y < 0$, the correct result is computed as follows:

$$\begin{aligned} X &= m_2(x_2-x_1) + x_2 \\ &\quad +m_1m_2\left|\frac{(x_1+x_3)}{2} - x_2\right|_{m_3} + M_L \\ &= m_2(x_2-x_1) + x_2 \\ &\quad +m_1m_2\left(\left|\frac{(x_1+x_3)}{2} - x_2\right|_{m_3} + \frac{m_3}{2}\right) \end{aligned} \tag{4.31}$$

The hardware realization of the proposed scheme is depicted by Figure 4.1. The implementation follows Equation (4.27) but the following should be noted.

At a first glance, D is a 3:1 adder. However, the extra input x_2 can be embedded into the partial product matrix of the m_1 multiplier according to the merged arithmetic principle. Furthermore, the modulo-$\frac{m_3}{2}$ operation associated with the adder C does not have to be explicitly computed. It can be replaced by at most one corrective addition.

In order to demonstrate that no explicit modulo operation is required by the proposed converter, we analyze the two possible extreme cases as follows:

Case 1: $(x_1 + x_3) = 0$ and $x_2 = 2n$. This results in the most negative value one may get. In this case Equation (4.27) reduces to $|-x_2|_{\frac{m_3}{2}}$. To perform the modulo $\frac{m_3}{2}$ operation, we need to do corrective additions. Given that $m_3+(-x_2) = (2n)+(-2n) = 2n-2n = 0$, for any positive even integer n, only one corrective addition with m_3 is required to compute the modulo.

Case 2: $(x_1 + x_3)$ is even and has the maximum possible value and x_2 is zero. This is the largest positive value one may get and Equation (4.27) reduces to $|\frac{(x_1+x_3)}{2}|_{\frac{m_3}{2}}$. Given that $m_3 - \frac{(x_1+x_3)}{2} = (2n) - \frac{(2n+1+2n-1)}{2} = 2n - 2n = 0$ the maximum sum in the modulo adder cannot exceed $\frac{m_3}{2}$, thus only one correction is required.

This means that the modulo $\frac{m_3}{2}$ operation can be implemented with at most one corrective addition.

As demonstrated by Equation (4.27), the final modulo-M also does not require explicit implementation.

The scheme is simplified by moving this addition before the m_1m_2 multiplication, hence transforming it into a corrective $\frac{m_3}{2}$ addition. As mentioned before, this correction must be applied when both of the following statements hold true:

$$x_1 > x_2 \tag{4.32}$$

$$\left|\frac{(x_1 + x_3)}{2} - x_2\right|_{m_3} = 0 \tag{4.33}$$

We now combine the modulo-$\frac{m_3}{2}$ operations by revisiting the correction rules:

- if tentative sum is smaller than -$\frac{m_3}{2}$ add m_3;
- if tentative sum is greater than or equal to -$\frac{m_3}{2}$ add $\frac{m_3}{2}$; Equation (4.33) holds true when the tentative sum is equal to $-m_3$, but we can see from *Case 1* that Equation (4.32) does not hold true, hence the extra $\frac{m_3}{2}$ addition is not needed;

Metrics	[2]	[44]	Our proposal
Area	4 adders 2 multipliers	3 adders 4 multipliers	4 adders 2 multipliers
Delay	3 additions 2 multiplications	2 additions 2 multiplications	3 additions 1 multiplication
Mod operations	$\frac{m_1}{2}m_3$	m_3	$\frac{m_3}{2}$

Table 4.8: Performance Comparison

- if tentative sum is zero and Equation (4.32) is true (the sign bit of adder A is 1) add $\frac{m_3}{2}$;
- otherwise do nothing;

In this way all modulo operations have been replaced by a single corrective addition or subtraction greatly reducing the complexity of the converter.

Figures 4.2 and 4.3 describe the hardware realization of the converters proposed in [44] and [2], respectively. The area, the delay, and the modulo operations required by the proposed converter and the one in [44] and [2] are summarized in Table 4.8. As one can observe in the table, our proposal requires less delay and operates on smaller magnitude operands with the same or less area. In particular, our proposal requires four 2:1 adders and two multipliers, Figure 4.2 requires one 3:1 adder, two 2:1 adders, and four multipliers while Figure 4.3 requires one 3:1 adder, three 2:1 adders and two multipliers. In terms of critical path delay, our scheme requires 3 additions and 1 multiplication with mod-n operations, the converter in [44] requires 2 additions and 2 multiplications with mod-$2n$ operations whereas the converter in [2] requires 3 additions, 2 multiplications with mod-$\frac{m_1}{2}m_3$ operations. Consequently, the new converter introduced in this section requires less delay with the same or less area. Moreover, due to the fact that the proposed scheme operates on smaller magnitude operands, it requires less complex adders and multipliers, which potentially results in even faster and smaller implementations.

Given that larger dynamic range is of practical interest, we present an efficient reverse converter for the moduli set $\{2n+3, 2n+2, 2n+1, 2n\}$, which is an extension of the $\{2n+2, 2n+1, 2n\}$ moduli set in the next section.

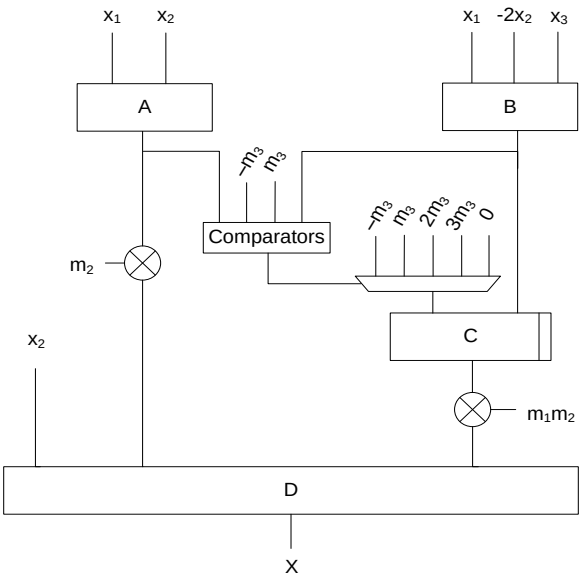


Figure 4.1: Hardware Realization of Our Proposal

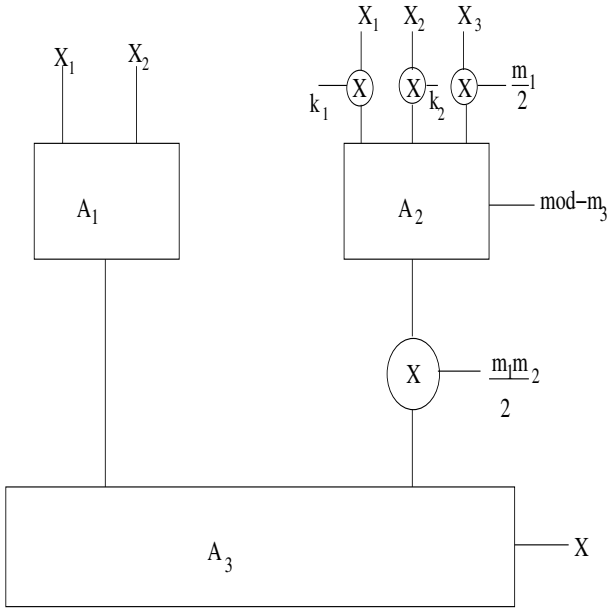


Figure 4.2: Converter Data Path for [44]

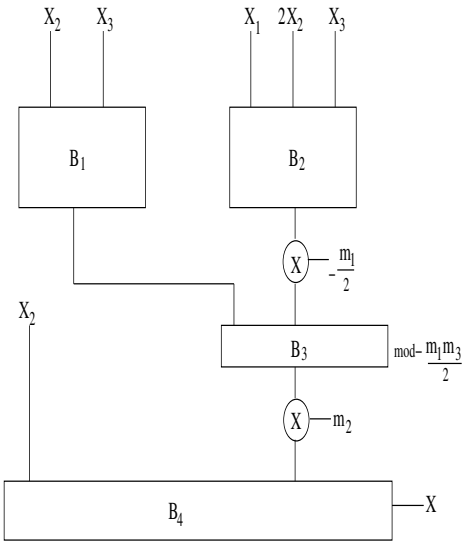


Figure 4.3: Converter Data Path for [2]

4.4 4-Moduli Set with a Common Factor

The main idea behind the proposed converter presented in this section is to simplify Equation (4.1) by eliminating the large modulo M and by removing the cost of computing M_i^{-1}. We demonstrate that the first one is possible for any 4-moduli RNS, while the second one can be achieved only for 4-moduli sets, which are not pairwise relatively prime. We first introduce a modified CRT for any moduli set of length four, which does not require mod-M computations.

Theorem 4.8 *For a moduli set $\{m_i\}_{i=1,4}$, $m_1 > m_2 > m_3 > m_4$, the decimal equivalent X of the residues (x_1, x_2, x_3, x_4) can be computed by using mod-m_4 (the smallest modulus) instead of the large mod-M operations as:*

$$\begin{aligned} X &= (x_1 + x_2 + x_3) + m_1 m_2 m_3 | k_1 x_1 + k_2 x_2 \\ &\quad + k_3 x_3 + \left| M_4^{-1} \right|_{m_4} x_4 |_{m_4}, \end{aligned} \tag{4.34}$$

where M_4^{-1} is the multiplicative inverse of M_4, $k_1 = \frac{(M_1 \left| M_1^{-1} \right|_{m_1} - 1)}{m_1 m_2 m_3}$, $k_2 = \frac{(M_2 \left| M_2^{-1} \right|_{m_2} - 1)}{m_1 m_2 m_3}$ and $k_3 = \frac{(M_3 \left| M_3^{-1} \right|_{m_3} - 1)}{m_1 m_2 m_3}$.

Proof:
We utilize the lemmas presented in [130]:
Lemma 1: $|am_1|_{m_1m_2} = m_1 |a|_{m_2}$
Lemma 2: $M_1 \left|M_1^{-1}\right|_{m_1} = 1 + k_1m_1m_2m_3$
Lemma 3: $M_2 \left|M_2^{-1}\right|_{m_2} = 1 + k_2m_1m_2m_3$
Lemma 4:$M_3 \left|M_3^{-1}\right|_{m_3} = 1 + k_3m_1m_2m_3$
Expanding Equation (4.1) for $t = 4$ we obtain:
$X = |M_1 \left|M_1^{-1}\right|_{m_1} x_1 + M_2 \left|M_2^{-1}\right|_{m_2} x_2$

$$+M_3 \left|M_3^{-1}\right|_{m_3} x_3 + M_4 \left|M_4^{-1}\right|_{m_4} x_4|_{m_1m_2m_3m_4} \tag{4.35}$$

Using Lemma 2 and 3 in the above equation, we obtain:
$X = |(1 + k_1m_1m_2m_3)x_1 + (1 + k_2m_1m_2m_3)x_2$

$$+(1 + k_3m_1m_2m_3)x_3 + M_4 \left|M_4^{-1}\right|_{m_4} x_4|_{m_1m_2m_3m_4} \tag{4.36}$$

Further simplification gives:
$X = (x_1 + x_2 + x_3) + |k_1m_1m_2m_3x_1 + k_2m_1m_2m_3x_2$

$$+k_3m_1m_2m_3x_3 + M_4 \left|M_4^{-1}\right|_{m_4} x_4|_{m_1m_2m_3m_4} \tag{4.37}$$

Applying Lemma 1, we get:

$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2$

$$+k_3x_3 + M_4^* \left|M_4^{-1}\right|_{m_4} x_4|_{m_4} \tag{4.38}$$

Here, $M_4^* = \frac{M_4}{m_1m_2m_3} = 1$, the above equation then reduces to:
$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2$

$$+k_3x_3 + \left|M_4^{-1}\right|_{m_4} x_4|_{m_4} \tag{4.39}$$

It can be observed that Equation (4.39) makes use of mod-m_4 (the smallest modulus) instead of mod-M operations, thus the magnitude of the involved

values is smaller than in both the traditional CRT and the new CRT [130], given that k_1, k_2 and k_3 can be precomputed.

The next simplification step is the elimination of the M_i^{-1}. To achieve that, we restrict to the new superset $\{2n+3, 2n+2, 2n+1, 2n\}$. Let this set be represented by $\{m_1, m_2, m_3, m_4\}$ where m_2 and m_4 have a common factor of 2. Suppose that the set $\{m_1, m_2, m_3, m_4\}$ is mapped into a set of pairwise relatively prime moduli set $\{\mu_1, \mu_2, \mu_3, \mu_4\}$, the new dynamic range will be given by:

$$M_L = \prod_{i=1}^{4} \mu_i, \tag{4.40}$$

where μ_i is any chosen set of integers within the given moduli. It should be noted that μ_i may have several sets. As discussed earlier in Section 4.1, for moduli set with a common factor, not every residue set corresponds to a number. Particularly, with the moduli set $\{2n+3, n+1, 2n+1, 2n\}$, even and odd numbers n that are multiples of 3 result into inconsistency. Thus, we choose sets of μ_is that are pairwise relatively prime for any even and odd integers n that are not multiples of 3. The valid sets for n-even and-odd are represented by $\{2n+3, n+1, 2n+1, 2n\}$ and $\{2n+3, n+1, 2n+1, 2n\}$, respectively. Next, we show that the computation of multiplicative inverses for these sets can be eliminated for any even or odd integers n, which are not multiples of 3 using the following theorem:

Theorem 4.9 *Given the moduli set* $\{2n+3, 2n+2, 2n+1, 2n\}$ *with* $m_1 = 2n+3, m_2 = 2n+2, m_3 = 2n+1, m_4 = 2n$*, for any even integer* $n > 0$*, the following hold true:*

$$\left|M_2^{-1}\right|_{m_2} = \frac{n}{2} + 1, \tag{4.41}$$

$$\left|M_3^{-1}\right|_{m_3} = 2n. \tag{4.42}$$

Proof:

If it can be demonstrated that $\left|\left(\frac{n}{2}+1\right) \times (m_1 m_3 m_4)\right|_{m_2} = 1$, then $\left(\frac{n}{2}+1\right)$ is the multiplicative inverse of $m_1 m_3 m_4$ with respect to m_2. $\left|\left(\frac{n}{2}+1\right) \times (m_1 m_3 m_4)\right|_{m_2}$ is given by: $\left|(2n+3)(2n+1)(2n)\left(\frac{n}{2}+1\right)\right|_{n+1} = |(n+1)(4n^3+12n^2+5n) + 2n^2 + n|_{n+1} = ||(n+1)(4n^3+12n^2+5n)|_{n+1} + |2n^2+n|_{n+1}|_{n+1} = |0+1|_{n+1} = 1$, thus Equation (4.41) holds true.

In the same way if $|(2n) \times (m_1 m_2 m_4)|_{m_3} = 1$, then $2n$ is the multiplicative inverse of $(m_1 m_2 m_4)$ with respect to m_3. $|(2n) \times (m_1 m_2 m_4)|_{m_3} = 1$ is given by: $|(2n+3)(n+1)(2n)(2n)|_{2n+1} = |(2n+3)(n+1)(4n^2)|_{2n+1} = |(4n^3 + 8n^2)(2n+1)+(4n^2)|_{2n+1} = ||(4n^3+8n^2)(2n+1)|_{2n+1}+|(4n^2)|_{2n+1}|_{2n+1} = |0+1|_{2n+1} = 1$, thus Equation (4.42) holds true.

Next, it can be shown that the multiplicative inverses of M_1 and M_4 also exist for any even integer $n > 0$ and is demonstrated by the following theorems:

Theorem 4.10 *For even numbers of the form* $\{2, 8, 14, 20, 26, 32, ...\}$*, represented by* $n = \{6k-4\}_{k=1,2,3,...}$

$$\left|M_1^{-1}\right|_{m_1} = 4k - 2, \tag{4.43}$$

$$\left|M_4^{-1}\right|_{m_4} = 2k - 1, \tag{4.44}$$

Proof:
From the Theorem, $|M_1^{-1}|_{m_1} = 4k - 2 = 2(2k-1)$. Thus, $|M_1^{-1}|_{m_1} = 2|M_4^{-1}|_{m_4}$, then we shall show the proof of $|M_1^{-1}|_{m_1}$ only. $M_1 = \frac{m_2}{2} m_3 m_4$, meaning that $M_1 = (n+1)(2n+1)(2n)$, for different values of n, $|M_1 M_1^{-1}|_{m_1}$ will be given by: $n = 2$, when $|M_1^{-1}|_{m_1} = 2$, and also $|2(n+1)(2n+1)(2n)|_{2n+3} = |4n(2n^2+3n+1)|_{2n+3} = |4n^2(2n+3)+4n|_{2n+3} = ||4n^2(2n+3)|_{2n+3} + |4n|_{2n+3}|_{2n+3} = |0+1|_{2n+3} = 1$.

Similarly, when $n = 8$, $|M_1^{-1}|_{m_1} = 6$, and $|6(n+1)(2n+1)(2n)|_{2n+3} = ||12n^2(2n+3)|_{2n+3} + |12n|_{2n+3}|_{2n+3} = |0+1|_{2n+3} = 1$.

Again, when $n = 14$, $|M_1^{-1}|_{m_1} = 10$, and $|10(n+1)(2n+1)(2n)|_{2n+3} = ||20n^2(2n+3)|_{2n+3} + |20n|_{2n+3}|_{2n+3} = |0+1|_{2n+3} = 1$.

Hence, if it is true for $n = 2, 8, 14$, then it will be true for any integer n in this category.

Theorem 4.11 *For even numbers of the form* $\{4, 10, 16, 22, 28, 34, ...\}$ *, represented by* $n = \{6k-2\}_{k=1,2,3,...}$

$$\left|M_1^{-1}\right|_{m_1} = 8k - 2, \tag{4.45}$$

$$\left|M_4^{-1}\right|_{m_4} = 10k - 3, \tag{4.46}$$

Proof It can be proved in a similar manner to Theorem 4.10.

It should be noted that for any n-even that are not multiples of 3, the following expressions can be deduced from Theorem 4.9:

$$\left|M_2^{-1}\right|_{m_2} = \frac{m_2+2}{4}, \left|M_3^{-1}\right|_{m_3} = m_4. \tag{4.47}$$

Using Equation (4.54) and by proper substitutions in Theorem 4.8, we can particularize it for 4-moduli RNS sharing a common factor as follows:

Corollary 4.2 *For the moduli set* $\{2n+3, 2n+2, 2n+1, 2n\}$*, the decimal equivalent* X *of the residues* (x_1, x_2, x_3, x_4) *can be computed as follows:*

1. ***(Using Theorem 4.10)**:*
   $$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2 + k_3x_3 + (2k-1)x_4|_{m_4}, \tag{4.48}$$
   where $k_1 = \frac{(m_2m_3m_4(4k-2)-1)}{m_1m_2m_3}$,
   $k_2 = \frac{\left(m_1m_3m_4\left(\frac{m_2+2}{4}\right)-1\right)}{m_1m_2m_3}$, *and*
   $k_3 = \frac{(m_1m_2m_4(m_4)-1)}{m_1m_2m_3}$
2. ***(Using Theorem 4.11)**:*
   $$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2 + k_3x_3 + (10k-3)x_4|_{m_4}, \tag{4.49}$$
   where $k_1 = \frac{(m_2m_3m_4(8k-1)-1)}{m_1m_2m_3}$,
   $k_2 = \frac{\left(m_1m_3m_4\left(\frac{m_2+2}{4}\right)-1\right)}{m_1m_2m_3}$ *and*
   $k_3 = \frac{(m_1m_2m_4(m_4)-1)}{m_1m_2m_3}$

Proof: Trivial with proper substitutions for the values of $\left|M_2^{-1}\right|_{m_2}$ and $\left|M_3^{-1}\right|_{m_3}$ together with $\left|M_1^{-1}\right|_{m_1}$ and $\left|M_4^{-1}\right|_{m_4}$, which are obtained from Theorems 4.10 and 4.11.

Since Theorem 4.9 holds true for any integer $n > 0$, we need only to show that the multiplicative inverses of M_1 and M_4 for any n-odd, which are multiples of 3, do exist and can be computed as demonstrated by the following theorems:

Theorem 4.12 *For odd numbers of the form* $\{5, 11, 17, 23, 29, 35, ...\}$*, represented by* $n = \{6q - 1\}_{q=1,2,3,...}$

$$\left|M_1^{-1}\right|_{m_1} = 4q, \tag{4.50}$$

$$\left|M_4^{-1}\right|_{m_4} = q, \tag{4.51}$$

Proof:
From the Theorem, $|M_1^{-1}|_{m_1} = 4q$. Thus, $|M_1^{-1}|_{m_1} = 4|M_4^{-1}|_{m_4}$, then we shall show the proof of $|M_1^{-1}|_{m_1}$ only. $M_1 = m_2 m_3 \frac{m_4}{2}$, meaning that $M_1 = (2n + 2)(2n + 1)(n)$, for different values of n, $|M_1 M_1^{-1}|_{m_1}$ will be given by: $n = 5$, when $|M_1^{-1}|_{m_1} = 4$, and also $|4(2n + 2)(2n + 1)(n)|_{2n+3} = |16n^3 + 24n^2 + 8n|_{2n+3} = ||8n^2(2n + 3)|_{2n+3} + |8n|_{2n+3}|_{2n+3} = |0 + 1|_{2n+3} = 1$.

Similarly, when $n = 11$, $|M_1^{-1}|_{m_1} = 8$, and $|8(2n + 2)(2n + 1)(n)|_{2n+3} = |32n^3 + 48n^2 + 16n|_{2n+3} = ||16n^2(2n + 3)|_{2n+3} + |16n|_{2n+3}|_{2n+3} = |0 + 1|_{2n+3} = 1$.

Again, when $n = 17$, $|M_1^{-1}|_{m_1} = 12$, and $|12(2n + 2)(2n + 1)(n)|_{2n+3} = |48n^3 + 72n^2 + 24n|_{2n+3} = ||24n^2(2n + 3)|_{2n+3} + |24n|_{2n+3}|_{2n+3} = |0 + 1|_{2n+3} = 1$.

Hence, if it is true for $n = 5, 11, 17$, then it will be true for any odd integer n in this category.

Theorem 4.13 *For odd numbers of the form* $\{7, 13, 19, 25, 31, ...\}$*, represented by* $n = \{6q + 1\}_{q=1,2,3,...}$

$$\left|M_1^{-1}\right|_{m_1} = 8q + 3, \tag{4.52}$$

$$\left|M_4^{-1}\right|_{m_4} = 5q + 1, \tag{4.53}$$

Proof:
We first show that $|M_1^{-1}|_{m_1}$ is valid as follows: $M_1 = m_2 m_3 \frac{m_4}{2} = (2n + 2)(2n+1)(n)$, for different values of n, $|M_1 M_1^{-1}|_{m_1}$ will be given by: $n = 7$, when $|M_1^{-1}|_{m_1} = 11$, and also $|11(2n+2)(2n+1)(n)|_{2n+3} = |44n^3+66n^2+22n|_{2n+3} = ||22n^2(2n + 3)|_{2n+3} + |22n|_{2n+3}|_{2n+3} = |0 + 1|_{2n+3} = 1$.

Similarly, when $n = 13$, $|M_1^{-1}|_{m_1} = 19$, and $|19(2n + 2)(2n + 1)(n)|_{2n+3} = |76n^3 + 114n^2 + 38n|_{2n+3} = ||38n^2(2n + 3)|_{2n+3} + |38n|_{2n+3}|_{2n+3} = |0 + 1|_{2n+3} = 1$.

Again, when $n = 19$, $|M_1^{-1}|_{m_1} = 27$, and $|27(2n+2)(2n+1)(n)|_{2n+3}$ $= |108n^3 + 162n^2 + 54n|_{2n+3} = ||54n^2(2n+3)|_{2n+3} + |54n|_{2n+3}|_{2n+3}$ $= |0+1|_{2n+3} = 1$.

Hence, if it is true for $n = 7, 13, 19$, then it will be true for any odd integer n in this category.

We then show the validity of $|M_4^{-1}|_{m_4}$ as follows: $M_4 = m_1m_2m_3 = (2n+3)(2n+2)(2n+1)$, for different values of n, $|M_4M_4^{-1}|_{m_4}$ will be given by: $n = 7$, when $|M_4^{-1}|_{m_4} = 6$, and also $|6(2n+3)(2n+2)(2n+1)|_n = |6(8n^3 + 24n^2 + 22n + 6)|_n = ||6(8n^2 + 24n + 22)(n)|_n + |36|_n|_n = |0+1|_n = 1$.

Similarly, when $n = 13$, $|M_4^{-1}|_{m_4} = 11$, and $|11(2n+3)(2n+2)(2n+1)|_n$ $= |11(8n^3 + 24n^2 + 22n + 6)|_n = ||11(8n^2 + 24n + 22)(n)|_n + |66|_n|_n$ $= |0+1|_n = 1$.

Again, when $n = 19$, $|M_4^{-1}|_{m_4} = 16$, and $|16(2n+3)(2n+2)(2n+1)|_n$ $= |16(8n^3 + 24n^2 + 22n + 6)|_n = ||16(8n^2 + 24n + 22)(n)|_n + |96|_n|_n$ $= |0+1|_n = 1$.

Hence, if it is true for $n = 7, 13, 19$, then it will be true for any odd integer n in this category.

Putting $m_2 = 2n+2$ and $m_4 = 2n$ in Equations (4.41) and (4.42), respectively, we obtain:

$$\left|M_2^{-1}\right|_{m_2} = \frac{m_2+2}{4}, \left|M_3^{-1}\right|_{m_3} = m_4. \tag{4.54}$$

Using Equation (4.54) and by proper substitutions in Theorem 4.8, we can particularize it for 4-moduli RNS sharing a common factor as follows:

Corollary 4.3 *For the moduli set* $\{2n+3, 2n+2, 2n+1, 2n\}$*, the decimal equivalent* X *of the residues* (x_1, x_2, x_3, x_4)*, for any* $n > 0$*, which are not multiples of* 3*, can be computed as follows:*

1. ***(Using Theorem 4.12):***

$$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2 + k_3x_3 + kx_4|_{m_4}, \tag{4.55}$$

where $k_1 = \frac{(4km_2m_3m_4-1)}{m_1m_2m_3}$,

$k_2 = \frac{\left(m_1m_3m_4\left(\frac{m_2+2}{4}\right)-1\right)}{m_1m_2m_3}$, *and*

$k_3 = \frac{(m_1m_2m_4(m_4)-1)}{m_1m_2m_3}$

2. ***(Using Theorem 4.13)***:
$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2$

$$+k_3x_3 + (5k+1)x_4|_{m_4}, \tag{4.56}$$

where $k_1 = \frac{(m_2m_3m_4(8k+3)-1)}{m_1m_2m_3}$,
$k_2 = \frac{\left(m_1m_3m_4\left(\frac{m_2+2}{4}\right)-1\right)}{m_1m_2m_3}$ *and*

$k_3 = \frac{(m_1m_2m_4(m_4)-1)}{m_1m_2m_3}$

Proof:
Trivial with proper substitutions for the values of $\left|M_2^{-1}\right|_{m_2}$ and $\left|M_3^{-1}\right|_{m_3}$ together with $\left|M_1^{-1}\right|_{m_1}$ and $\left|M_4^{-1}\right|_{m_4}$, which are obtained from Theorems 4.12 and 4.13.

Clearly, it can be seen that the numbers involved in the multiplication are very small when compared to the numbers involved in both the direct traditional CRT and the New CRT [130] implementations. Additionally, the large modulo M calculations are replaced by modulo calculations with the smallest modulus in the moduli set under consideration.

We take note here that in Table 4.9, mod Optns stands for Modulo Operations. As indicated in Table 4.9, in terms of area, the proposed converter requires the same area with the New CRT [130] whereas the traditional CRT [111] utilizes lesser area when compared to both the New CRT [130] and our approach. On the other hand, in terms of critical path delay, the CRT [111] requires 1 multiplication lesser than both the New CRT [130] and our technique but more important for the hardware complexity, the operands magnitude is significantly reduced by our proposal. More specifically, the modulo operation has been reduced from modulo $M = m_1m_2m_3m_4$ in [111] or $M = m_2m_3m_4$ in [130] to modulo $M = m_4$ in our scheme. This implies that our technique is manipulating smaller numbers when compared to the other techniques. The smaller the involved numbers in the calculation, the faster the arithmetic operations. Thus, our proposal is faster than the other techniques.

Finally, when compared to the existing similar 3-moduli set [2, 44, 98], the newly introduced four moduli set offers a larger dynamic range and a higher parallelism, which makes it more attractive for high performance computing.

In the next section, we show that Theorems 4.1 and 4.8 can be generalized such that they become applicable to any n-moduli set $m_{ii=1,k}$.

Metrics	CRT [111]	New CRT [130]	Our proposal
Area	1 adder 4 multipliers	1 adder 5 multipliers	1 adder 5 multipliers
Dely	1 addition 1 multiplication	1 addition 2 multiplications	1 addition 2 multiplications
Mod Optns	$m_1m_2m_3m_4$	$m_2m_3m_4$	m_4

Table 4.9: Performance Comparison

4.5 Modified Chinese Remainder Theorem

In the last section, we present Theorem 4.8, which is applicable for any 4-moduli set. We wish to show that this is also possible for 5-moduli set and then generalize it for any n-moduli set. This is seen as a big time improvement as it eliminates the burden of the large modulo M present in the traditional CRT.

Theorem 4.14 *For a moduli set $\{m_i\}_{i=1,5}$, $m_1 > m_2 > m_3 > m_4 > m_5$, the decimal equivalent X of the residues $(x_1, x_2, x_3, x_4, x_5)$ can be computed by using mod-m_5 (the smallest modulus) instead of the large mod-M operations as:*

$$\begin{aligned} X &= (x_1 + x_2 + x_3 + x_4) + m_1m_2m_3m_4|k_1x_1 + k_2x_2 \\ &\quad + k_3x_3 + k_4x_4 + \left|M_5^{-1}\right|_{m_5} x_5|_{m_5}, \end{aligned} \tag{4.57}$$

where M_5^{-1} is the multiplicative inverse of M_5, $k_1 = \frac{\left(M_1\left|M_1^{-1}\right|_{m_1} - 1\right)}{m_1m_2m_3m_4}$, $k_2 = \frac{\left(M_2\left|M_2^{-1}\right|_{m_2} - 1\right)}{m_1m_2m_3m_4}$, $k_3 = \frac{\left(M_3\left|M_3^{-1}\right|_{m_3} - 1\right)}{m_1m_2m_3m_4}$, and $k_4 = \frac{\left(M_4\left|M_4^{-1}\right|_{m_4} - 1\right)}{m_1m_2m_3m_4}$.

Proof:
We utilize the lemmas presented in [130]:

Lemma 1: $|am_1|_{m_1m_2} = m_1\,|a|_{m_2}$

Lemma 2: $M_1\left|M_1^{-1}\right|_{m_1} = 1 + k_1m_1m_2m_3m_4$

Lemma 3: $M_2\left|M_2^{-1}\right|_{m_2} = 1 + k_2m_1m_2m_3m_4$

Lemma 4:$M_3\left|M_3^{-1}\right|_{m_3} = 1 + k_3m_1m_2m_3m_4$

Lemma 5:$M_4\left|M_4^{-1}\right|_{m_4} = 1 + k_4m_1m_2m_3m_4$

Expanding Equation (4.1) for $t = 5$ we obtain: $X = |M_1 \left|M_1^{-1}\right|_{m_1} x_1 + M_2 \left|M_2^{-1}\right|_{m_2} x_2$

$$+M_3 \left|M_3^{-1}\right|_{m_3} x_3 + M_4 \left|M_4^{-1}\right|_{m_4} x_4 + M_5 \left|M_5^{-1}\right|_{m_5} x_5|_M \tag{4.58}$$

Putting Lemmas 2, 3, 4, and 5 in the above equation, we obtain:

$$X = |(1 + k_1M_5)x_1 + (1 + k_2M_5)x_2 + (1 + k_3M_5)x_3 + (1 + k_4M_5)x_4 + M_5 \left|M_5^{-1}\right|_{m_5} x_5|_M \tag{4.59}$$

Further simplification gives:

$$X = (x_1 + x_2 + x_3 + x_4) + |k_1M_5x_1 + k_2M_5x_2 + k_3M_5x_3 + k_4M_5x_4 + M_5 \left|M_5^{-1}\right|_{m_5} x_5|_{m_1m_2m_3m_4m_5} \tag{4.60}$$

Applying Lemma 1, we get:

$$X = (x_1 + x_2 + x_3) + m_1m_2m_3|k_1x_1 + k_2x_2 + k_3x_3 + k_4x_4 + M_5^* \left|M_5^{-1}\right|_{m_5} x_5|_{m_5} \tag{4.61}$$

Here, $M_5^* = \frac{M_5}{m_1m_2m_3m_4} = 1$, the above equation then reduces to:

$$X = (x_1 + x_2 + x_3 + x_4) + m_1m_2m_3m_4|k_1x_1 + k_2x_2 + k_3x_3 + k_4x_4 + \left|M_5^{-1}\right|_{m_5} x_5|_{m_5} \tag{4.62}$$

Next, we present a modified CRT for general n-moduli set, which does not require mod-M computations.

Theorem 4.15 *For a moduli set $\{m_i\}_{i=1,k}$, $m_1 > m_2 > m_3... > m_k$, the decimal equivalent X of the residues $(x_1, x_2, x_3, ..., x_k)$ can be computed by using mod-m_k (the smallest modulus) instead of the large mod-M operations as:*

$$X = \sum_{i=1}^{k-1} x_i + \prod_{i=1}^{k-1} m_i \left| \sum_{i=1}^{k-1} k_i x_i + \left|M_k^{-1} x_k\right|_{m_k} \right|_{m_k}, \tag{4.63}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i}$, M_i^{-1} is the multiplicative inverse of M_i with respect to m_i, and $k_i = \frac{\left(M_i \left|M_i^{-1}\right|_{m_i} - 1\right)}{\prod_{i=1}^{k-1} m_i}$.

Proof:
If Theorems 4.1, 4.8, and 4.14 hold true, then Theorem 4.15 also holds true.

4.6 Conclusion

This chapter resolved a number of issues regarding the utilization of moduli sets with common factors in building effective reverse converters. First, we assumed a general $\{m_i\}_{i=1,3}$ moduli set with the dynamic range $M = \prod_{i=1}^{k} m_i$ and introduced a modified CRT that requires mod-m_3 instead of mod-M calculations. This scheme can be utilized in conjunction with well established moduli sets, e.g, $\{2^n + 1, 2^n, 2^n - 1\}$ and $\{2n + 2, 2n + 1, 2n\}$ and makes the CRT based conversion more effective as it reduces the magnitude of the values involved in the conversion, thus the associated costs in area and delay. Subsequently, we further simplified the conversion process by focussing on $\{2n + 2, 2n + 1, 2n\}$ moduli set, which has a common factor of 2. Given that for such a moduli set, CRT cannot be directly applied, we introduced in a formal way a CRT based approach for this case, which requires the conversion of $\{2n + 2, 2n + 1, 2n\}$ set into moduli set with relatively prime moduli, i.e., $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, when n is even, $n \geq 2$ and $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, when n is odd, $n \geq 3$. We demonstrated that the moduli set transformation can be easily done. We further simplified the 3-moduli CRT for the specific case of $\{2n + 2, 2n + 1, 2n\}$ moduli set. For this case, the proposed CRT requires 4 additions, 4 multiplications and all the operations are mod-m_3 in case n is even and mod-$\frac{m_3}{2}$ if n is odd.

Second, we proposed another new RNS to decimal converter for the same moduli set $\{2n + 2, 2n + 1, 2n\}$ for both even and odd integer $n > 0$. We simplified the traditional CRT to obtain a reverse conveter that uses mod-n instead of mod-$(2n + 2)(2n)$ and mod-$2n$ required by other state of the art equivalent converters. Next, we presented a low complexity implementation that does not require the explicit use of the modulo operation in the conversion process as it is normally the case in the traditional CRT and other state of the art equivalent converters. In terms of area, our proposal requires four 2:1 adders and 2 multipliers while the best state of the art equivalent converter requires one 3:1 adder, two 2:1 adders, and four multipliers. In terms of critical path delay, the proposed scheme requires 3 additions and 1 multiplication with mod-n operations whereas the best state of the art equivalent converter requires 2 additions and 2 multiplications with mod-$2n$ operations.

Third, we proposed a new 4-moduli set $\{2n + 3, 2n + 2, 2n + 1, 2n\}$ that increases the dynamic range and the processing parallelism enabling efficient reverse conversion. We assume a general 4-moduli set $\{m_i\}_{i=1,4}$, $m_1 > m_2 > m_3 > m_4$, with the dynamic range $M = \prod_{i=1}^{4} m_i$ and introduced a modified CRT that requires mod-m_4 instead of mod-M calculations. This scheme also

can be utilized in conjunction with other well established $4-$moduli sets, e.g, $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ [121], $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$ [13], $\{2^n-3, 2^n+1, 2^n-1, 2^n+3\}$ [20]. Moreover, we further simplified the conversion process by focussing on $\{2n+3, 2n+2, 2n+1, 2n\}$ moduli set, which has a common factor of 2. Given that for such a moduli set CRT cannot be directly applied, we introduced in a formal way, just as we did in the previous cases, a CRT based approach for this case, which requires the conversion of $\{2n+3, 2n+2, 2n+1, 2n\}$ set into moduli set with relatively prime moduli, i.e., $\{m_1, \frac{m_2}{2}, m_3, m_4\}$ and $\{m_1, m_2, m_3, \frac{m_4}{2}\}$, when n is even and odd, respectively, which are not multiples of 3. We demonstrated that the moduli set transformation can be easily done. For this case, the proposed CRT requires the same or slightly larger area when compared to other existing techniques but all the operations are mod-m_4. This outperforms state of the art CRTs in terms of the magnitude of the numbers involved in the calculation and due to this fact, our proposal results in less complex adders and multipliers, which potentially results in even faster and smaller implementations. When compared to the existing similar 3-moduli set [2,44,98], this newly introduced four moduli set offers a larger dynamic range and a higher parallelism, which makes it more attractive for high performance computing. This proposal is particularly suitable in DSP applications where the moduli sets are restricted and the dynamic range does not necessarily need to be too large.

Finally, given the general $\{m_i\}_{i=1,k}$ moduli set with the dynamic range $M = \prod_{i=1}^{k} m_i$, we present a modified CRT that requires mod-m_k instead of mod-M calculations. This proposal is better than the existing CRTs in terms of conversion delay since the numbers involved in the calculations are smaller, which results in less complex adders and multipliers.

In the chapter to follow, we investigate moduli set which contains pairwise relatively prime moduli. For this class of moduli set, unlike moduli set with a common factor, no transformation is required and the entire dynamic range of the moduli set is utilized.

Chapter 5

Modulo Operation free Computation for the Moduli Set $\{2n+1, 2n, 2n-1\}$

In Chapter 4, we presented efficient data conversion techniques for moduli sets with common factors. It was shown that those moduli sets must be mapped to the moduli sets, which are relatively prime. Such a transformation is not required for the moduli set $\{2n+1, 2n, 2n-1\}$, which is investigated in this chapter. Apart from the transformation problem, the converters presented in Chapter 4 reduce the dynamic range of the original moduli set by 1-bit. Due to these reasons, we present an efficient Residue Number System (RNS) to Decimal converter for the moduli set $\{2n+1, 2n, 2n-1\}$ in this chapter.

This chapter is organized as follows. Section 5.1 introduces the necessary background for the $\{2n+1, 2n, 2n-1\}$ moduli set, Section 5.2 describes the proposed conversion algorithm, Section 5.3 describes the hardware implementation of the proposed algorithm. In Section 5.4, the performance of the proposed algorithm is evaluated while the chapter is concluded in Section 5.5.

5.1 Background

As discussed in Chapter 4, the residue number $(x_1, x_2, x_3, ..., x_k)$ for the moduli set $\{m_1, m_2, m_3, ..., m_k\}$ with the dynamic range $M = \prod_{i=1}^{k} m_i$, can be converted into the decimal number X, according to the Chinese Reminder

Theorem (CRT), as follows [111]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M , \tag{5.1}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i. Equation (5.1) can be simplified when certain moduli sets of interests like $\{2n+1, 2n, 2n-1\}$ are utilized for which specific converters have been designed with the following relations for $(x_1 + x_3)$ even and odd, respectively, in [99]:

$$X = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M \tag{5.2}$$

and

$$X = \left| \frac{M}{2} + \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M \tag{5.3}$$

The following relation, which uses smaller multipliers when compared to Equations (5.2) and (5.3) have also been presented in [2]:

$$\begin{aligned} X &= x_2 + m_2 \left| \left\lfloor \frac{(x_1 - x_3) + 2 z_0 n}{2} \right\rfloor \right. \\ &+ 2n \left. \left\lfloor \frac{(x_1 - 2x_2 + x_3) + 2 z_0 n}{2} \right\rfloor \right|_{m_1 m_3} , \end{aligned} \tag{5.4}$$

where z_0 is the XOR over the least significant bits of x_1 and x_3.

In the following section, we assume the same moduli set $\{2n+1, 2n, 2n-1\}$ and we introduce an RNS to decimal converter based on Equation (5.1) by eliminating the computation of the required multiplicative inverses. By doing that, we obtain exactly the same relations as Equations (5.2) and (5.3). Further, we simplify these expressions such that we obtain relations that use smaller multipliers and require lesser number of arithmetic operations when compared to Equation (5.4).

5.2 Modulo $(2n-1)$ Reverse Converter

Given the RNS number (x_1, x_2, x_3) for the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+1, 2n, 2n-1\}$, the proposed algorithm computes the decimal equivalent of this RNS number based on a further simplification of the well-known traditional CRT. First, we demonstrate that the computation of the multiplicative inverses can be eliminated. We further simplify the resulting CRT

to get a low complexity implementation that does not require explicit use of modulo operation at the final stage of computation.

Theorem 5.1 *Given the moduli set* $\{2n+1, 2n, 2n-1\}$ *with* $m_1 = 2n+1, m_2 = 2n, m_3 = 2n-1$ *the following hold true:*

$$|(m_1 m_2)^{-1}|_{m_3} = n, \tag{5.5}$$

$$|(m_2 m_3)^{-1}|_{m_1} = n+1, \tag{5.6}$$

$$|(m_1 m_3)^{-1}|_{m_2} = 2n-1. \tag{5.7}$$

Proof:
If it can be demonstrated that $|n \times (m_1 m_2)|_{m_3} = 1$, then n is the multiplicative inverse of $(m_1 m_2)$ with respect to m_3. $|n \times (m_1 m_2)|_{m_3}$ is given by: $|(2n+1)(2n)(n)|_{2n-1} = |(4n^2+2n)n|_{2n-1} = ||4n^3|_{2n-1} + |2n^2|_{2n-1}|_{2n-1} = |\frac{1}{2} + \frac{1}{2}|_{2n-1} = 1$, thus Equation (5.5) holds true.

In the same way if $|(n+1) \times (m_2 m_3)|_{m_1} = 1$, then $n+1$ is the multiplicative inverse of $(m_2 m_3)$ with respect to m_1. $|(n+1) \times (m_2 m_3)|_{m_1}$ is given by: $|2n(2n-1)(n+1)|_{2n+1} = |2n^2(2n+1) - 2n|_{2n+1} = ||2n^2(2n+1)|_{2n+1} + |-2n|_{2n+1}|_{2n+1} = |0+1|_{2n+1} = 1$, thus Equation(5.6) holds true.

Again, if $|(2n-1) \times (m_1 m_3)|_{m_2} = 1$, then $2n-1$ is the multiplicative inverse of $(m_1 m_3)$ with respect to m_2. $|(2n-1) \times (m_1 m_3)|_{m_2}$ is given by: $|(2n+1)(2n-1)(2n-1)|_{2n} = |8n^3 - 4n^2 - 2n + 1|_{2n} = ||2n(4n^2 - 2n - 1)|_{2n} + |1|_{2n}|_{2n} = |0+1|_{2n} = 1$, thus Equation (5.7) holds true.

The following theorem introduces a simplified way to compute the decimal equivalent of the RNS number.

Theorem 5.2 *The decimal equivalent of the RNS number* (x_1, x_2, x_3) *with respect to the moduli set* $\{m_1, m_2, m_3\}$ *in the form* $\{2n+1, 2n, 2n-1\}$ *is computed for* $(x_1 + x_3)$ *even and odd, respectively, as follows:*

$$X = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M \tag{5.8}$$

$$X = \left| \frac{M}{2} + \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_M \tag{5.9}$$

Proof:
Equation (5.1) for $k = 3$ is given by:

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M . \tag{5.10}$$

By substituting the multiplicative inverse values in Theorem (5.1) into Equation (5.10) we obtain the following:

$$\begin{aligned} X &= \left| (m_2 m_3)(\frac{m_2}{2} + 1)x_1 + (m_1 m_3)(m_3)x_2 + (m_1 m_2)(\frac{m_2}{2} x_3) \right|_M \\ &= \left| (m_2 m_3)(\frac{m_1 + 1}{2})x_1 + (m_1 m_3)(m_2 - 1)x_2 + (m_1 m_2)(\frac{m_3 + 1}{2})x_3 \right|_M \\ &= \left| (\frac{M}{2})x_1 + (\frac{m_2 m_3}{2})x_1 - m_1 m_3 x_2 + (\frac{M}{2})x_3 + (\frac{m_1 m_2}{2})x_3 \right|_M \\ &= \left| \left| (x_1 + x_3)\frac{M}{2} \right|_M + \left| (\frac{m_2 m_3}{2})x_1 \right|_M - |m_1 m_3 x_2|_M \right. \\ &\quad \left. + \left| (\frac{m_1 m_2}{2})x_3 \right|_M \right|_M \end{aligned} \tag{5.11}$$

From Equation (5.11), the following cases may be considered:

- If $(x_1 + x_3)$ is even then $|(x_1 + x_3)\frac{M}{2}|_M = 0$.
- If $(x_1 + x_3)$ is odd then $|(x_1 + x_3)\frac{M}{2}|_M = \frac{M}{2}$.

The first case is trivial while we show the validity of the second case as follows. The term (x_1+x_3)-odd can be perceived in two ways. It is either $(x_1+x_3) = 1$ or $(x_1+x_3) > 1$. If $(x_1+x_3) = 1$, $\frac{M}{2}$ will always be less than M, thus, the term $\left|\frac{M}{2}\right|_M = \frac{M}{2}$ is always true. On the other hand, suppose $(x_1 + x_3) = a > 1$, where a is an odd integer. Then a can be expressed in terms of any even integer b as: $a = b + 1$. Therefore, the second case $(x_1 + x_3) > 1$-odd can be written as:

$$\begin{aligned} \left| (x_1 + x_3)\frac{M}{2} \right|_M &= \left| (1 + b)\frac{M}{2} \right|_M = \left| \frac{M}{2} + b\frac{M}{2} \right|_M \\ &= \left| \left| \frac{M}{2} \right|_M + \left| b\frac{M}{2} \right|_M \right|_M = \left| \frac{M}{2} + 0 \right|_M \\ &= \left| \frac{M}{2} \right|_M = \frac{M}{2} \end{aligned}$$

Hence, for $(x_1 + x_3)$-odd, the term $\left|(x_1 + x_3)\frac{M}{2}\right|_M = \frac{M}{2}$ is always true. Given that the above two cases are true, Equation (5.11) can be re-written for the first and the second case, respectively, as:

$$X = \left|\frac{m_2m_3}{2}x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_M \tag{5.12}$$

$$X = \left|\frac{M}{2} + \frac{m_2m_3}{2}x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_M \tag{5.13}$$

Thus, Equation (5.8) and (5.9) hold true. Equations (5.8) and (5.9) are exactly the same as the ones derived based on a weighting function in [99] and [100]. Thus, the hardware realization based on these equations is the same as the ones presented in [99] and [100].

We propose to further simplify Equations (5.8) and (5.9) using the following theorem:

Theorem 5.3 *Given the RNS number (x_1, x_2, x_3) for the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+1, 2n, 2n-1\}$, the decimal equivalent of this RNS number is computed for $(x_1 + x_3)$ even and odd, respectively, as follows:*

$$X = \left|-m_2x_1 + m_1\left|\frac{m_2}{2}(x_1 + x_3) - m_3x_2\right|_{m_2m_3}\right|_M \tag{5.14}$$

$$X = |-m_2x_1 + m_1\left|\frac{m_2m_3}{2} + \frac{m_2}{2}(x_1 + x_3) - m_3x_2\right|_{m_2m_3}\Big|_M \tag{5.15}$$

Proof:
To prove this theorem we use the following lemma presented in [131]:

$$|am_1|_{m_1m_2} = m_1\,|a|_{m_2} \tag{5.16}$$

From Equation (5.8), we have

$$\begin{aligned} X &= \left|\frac{m_2m_3}{2}x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_{m_1m_2m_3} \\ &= \left|\frac{m_2}{2}(m_1 - 2)x_1 - m_1m_3x_2 + \frac{m_1m_2}{2}x_3\right|_{m_1m_2m_3} \\ &= x_1 + \left|\frac{m_1m_2}{2}(x_1 + x_3) - m_1x_1 - m_1m_3x_2\right|_{m_1m_2m_3} \end{aligned} \tag{5.17}$$

Applying Equation (5.16) we obtain

$$X = \left| x_1 + m_1 \left| \frac{m_2}{2}(x_1 + x_3) - x_1 - m_3 x_2 \right|_{m_2 m_3} \right|_M , \tag{5.18}$$

which can be further simplified as:

$$X = \left| -m_2 x_1 + m_1 | \frac{m_2}{2}(x_1 + x_3) - m_3 x_2 |_{m_2 m_3} \right|_M \tag{5.19}$$

thus Equation (5.14) holds true.

From Equation (5.9), we have

$$\begin{aligned} X &= \Big| \frac{m_1 m_2 m_3}{2} + \frac{m_2 m_3}{2} x_1 \\ &\quad - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \Big|_{m_1 m_2 m_3} \\ &= \Big| \frac{m_1 m_2 m_3}{2} + \frac{m_2}{2}(m_1 - 2) x_1 \\ &\quad - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \Big|_{m_1 m_2 m_3} \\ &= x_1 + \Big| \frac{m_1 m_2 m_3}{2} + \frac{m_1 m_2}{2}(x_1 + x_3) \\ &\quad - m_1 x_1 - m_1 m_3 x_2 |_{m_1 m_2 m_3} \end{aligned} \tag{5.20}$$

Applying Equation (5.16) we obtain

$$\begin{aligned} X &= |x_1 \\ &\quad + m_1 \left| \frac{m_2 m_3}{2} + \frac{m_2}{2}(x_1 + x_3) - x_1 - m_3 x_2 \right|_{m_2 m_3} \Big|_M , \end{aligned} \tag{5.21}$$

which can be further simplified as

$$\begin{aligned} X &= \Big| -m_2 x_1 + m_1 \Big| \frac{m_2 m_3}{2} \\ &\quad + \frac{m_2}{2}(x_1 + x_3) - m_3 x_2 \Big|_{m_2 m_3} \Big|_M \end{aligned} \tag{5.22}$$

thus Equation (5.15) holds also true. We propose to further simplify Equations (5.14) and (5.15) in order to obtain a converter that only utilizes modulo-m_3. This is achieved by the following two theorems.

Theorem 5.4 *Given the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+1, 2n, 2n-1\}$, the decimal equivalent of this RNS number is computed for $(x_1 + x_3)$ even as follows:*

$$\begin{cases} X = y, & y \geq 0 \\ X = y + M, & y < 0 \end{cases} \tag{5.23}$$

where

$$\begin{aligned} y &= m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \end{aligned} \tag{5.24}$$

Proof:
To prove this theorem we use the following lemma presented in [131]:

$$|am_1|_{m_1 m_2} = m_1 \, |a|_{m_2} \tag{5.25}$$

From Equation (5.14), we have

$$\begin{aligned} X &= |-m_2 x_1 \\ &\quad + m_1 \left| \frac{m_2}{2}(x_1 + x_3) - m_3 x_2 \right|_{m_2 m_3} \Big|_M \end{aligned} \tag{5.26}$$

Substituting $m_3 = m_2 - 1$, we get

$$\begin{aligned} X &= |-m_2 x_1 \\ &\quad + m_1 \left| \frac{m_2}{2}(x_1 + x_3) - x_2(m_2 - 1) \right|_{m_2 m_3} \Big|_M \\ &= |-m_2 x_1 + m_1 x_2 \\ &\quad + m_1 \left| m_2 \left(\frac{(x_1 + x_3)}{2} - x_2 \right) \right|_{m_2 m_3} \Big|_M \end{aligned} \tag{5.27}$$

Subsequently, we apply Equation (5.25) and obtain

$$\begin{aligned} X &= |-m_2 x_1 + x_2(m_2 + 1) \\ &\quad + m_1 m_2 \left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \Big|_M \end{aligned}$$

which can be further simplified as

$$\begin{aligned} X &= |m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \Big|_M \end{aligned} \tag{5.28}$$

Equation (5.28) is the general expression of Equation (5.24), valid for both y positve and negative. The next stage of the proof is to demonstrate that only one corrective addition is required for the calculation of the mod-M operation. We demonstrate that by considering the the most positive value one may get in (5.28).

- *Most positive value*: in order to get the most positive value in (5.28), the following must hold true: $\left| \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = m_3 - 1$, $x_1 = m_1 - 1$, $x_2 = 1$, $x_3 = m_3 - 1$. Substituting all these values in Equation (5.28), we obtain

$$\begin{aligned} X &= |m_2(1 - m_1 + 1) + 1 + m_1 m_2 (m_3 - 1)|_M \\ &= |2m_2 - m_1 m_2 + 1 + M - m_1 m_2|_M \\ &= |M - 2m_1 m_2 + 2m_2 + 1|_M \end{aligned} \tag{5.29}$$

Since $0 < M - 2m_1 m_2 + 2m_2 + 1 < M$, no corrective addition of M is required in order to obtain the desired result.

On the other hand, for $y < 0$, the following must hold true: $\left| \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = 0$, $x_1 > x_2$. We demonstrate that only one corrective addition is required in order to compute the correct result. This is achieved by computing the most negative result one may have in Equation (5.28).

- *Most negative value*: in order to get the most negative value in Equation (5.28), we substitute $x_1 = m_1 - 1$, $x_2 = 0$, and $\left| \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = 0$ in Equation (5.28) and we obtain

$$\begin{aligned} X &= |m_2(0 - m_1 + 1)|_M \\ &= |-m_1 m_2 + 1|_M \end{aligned} \tag{5.30}$$

Since $0 < -m_1 m_2 + 1 + M < M$, only one corrective addition is therefore required in order to obtain the correct result if $y < 0$. Thus, (5.23) holds true.

For the case when $y < 0$, the correct result can be computed as follows:

$$\begin{aligned} X &= m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left(\left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} + m_3 \right) \end{aligned} \tag{5.31}$$

Theorem 5.5 *Given the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+1, 2n, 2n-1\}$, the decimal equivalent of this RNS number is computed for $(x_1 + x_3)$ odd as follows:*

$$\begin{cases} X = y, & y \geq 0 \\ X = y + M, & y < 0 \end{cases} \tag{5.32}$$

where

$$\begin{aligned} y &= m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left| \frac{m_3}{2} + \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \end{aligned} \tag{5.33}$$

Proof:
Similarly, from Equation (5.15), we have

$$\begin{aligned} X &= |-m_2 x_1 \\ &\quad + m_1 \left| \frac{m_2 m_3}{2} + \frac{m_2}{2}(x_1 + x_3) - m_3 x_2 \right|_{m_2 m_3} \Bigg|_M \end{aligned} \tag{5.34}$$

Applying the equation $m_3 = m_2 - 1$, we get

$$\begin{aligned} X &= |-m_2 x_1 \\ &\quad + m_1 \left| \frac{m_2 m_3}{2} + \frac{m_2}{2}(x_1 + x_3) - x_2(m_2 - 1) \right|_{m_2 m_3} \Bigg|_M \\ &= |-m_2 x_1 + m_1 x_2 \\ &\quad + m_1 \left| m_2 \left(\frac{m_3}{2} + \frac{(x_1 + x_3)}{2} - x_2 \right) \right|_{m_2 m_3} \Bigg|_M \end{aligned}$$

Applying Equation (5.25) we obtain

$$\begin{aligned} X &= |-m_2 x_1 + x_2(m_2 + 1) \\ &\quad + m_1 m_2 \left| \frac{m_3}{2} + \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \Bigg|_M \end{aligned} \tag{5.35}$$

which can be further simplified as

$$\begin{aligned} X &= |m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left| \frac{m_3}{2} + \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} \Bigg|_M \end{aligned} \tag{5.36}$$

Again, from Equation (5.33), it can be seen easily that Equation (5.36) is the same as $|y|_M$. Just as earlier described, we need to demonstrate that only one corrective addition is required for the calculation of mod-M operation. We demonstrate that by considering the most positive value one may get in Equation (5.36).

- *Most positive value*: in order to get the most positive value in Equation (5.36), the following must hold true: $\left| \frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = m_3 - 1$, $x_1 = 1$, $x_2 = 1$, $x_3 = m_3 - 1$.

Substituting the above values in Equation (5.36), we obtain

$$\begin{aligned} X &= |1 + m_1 m_2 (m_3 - 1)|_M \\ &= |M - m_1 m_2 + 1|_M \end{aligned} \tag{5.37}$$

Since $0 < M - m_1 m_2 + 1 < M$, no corrective addition is required for the calculation of mod-M.

Again, for $y < 0$, the following must hold true: $\left| \frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = 0$, $x_1 > x_2$. We demonstrate that only one corrective addition is needed in order to obtain the correct result in Equation (5.36). This is achieved by considering the most negative value one may get in Equation (5.36).

- *Most negative value*: in order to get the most negative value in Equation (5.36), the following must hold true: given that $\left| \frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2 \right|_{m_3} = 0$, $x_1 = 1$, $x_2 = 0$, and $x_3 = m_3 - 1$. Substitute these values in Equation (5.36), we obtain

$$\begin{aligned} X &= |m_2(0 - 1) + 1 + 0|_M \\ &= |-m_2 + 1|_M \end{aligned} \tag{5.38}$$

Since $0 < -m_2 + 1 + M < M$, only one corrective addition is required in order to obtain correct result. For the case when $y < 0$, the correct result can

be computed as follows:

$$\begin{aligned} X &= m_2(x_2 - x_1) + x_2 \\ &\quad + m_1 m_2 \left(\left| \frac{m_3}{2} + \frac{(x_1 + x_3)}{2} - x_2 \right|_{m_3} + m_3 \right) \end{aligned} \tag{5.39}$$

In order to illustrate the way the proposed converter works, we present in the remainder of this section two conversion examples.

Example: Let us assume RNS$(7|6|5)$ and the integer number $X = 20$. Its RNS representation can be obtained as follows:

$$x_1 = |20|_7 = 6, x_2 = |20|_6 = 2, x_3 = |20|_5 = 0.$$

Thus, $20 = (6, 2, 0)_{RNS(7|6|5)}$ and similarly another number $11 = (4, 5, 1)_{RNS(7|6|5)}$. Now, let us convert the RNS numbers back to Decimal using the proposed schemes:

- $(6, 2, 0)_{RNS(7|6|5)}$. Here, $x_1 = 6$, $m_1 = 7$, $x_2 = 2$, $m_2 = 6$, $x_3 = 0$, $m_3 = 5$. It can be seen that $x_1 + x_3$ is even and $\left|\frac{6}{2} - 2\right|_5$ is not equal to zero, thus we employ Equation (5.24) as follows:

$$\begin{aligned} X &= 6(2-6) + 2 + 42 \left| \frac{6}{2} - 2 \right|_5 \\ &= -24 + 2 + 42 \\ &= 20, \end{aligned}$$

  as it should.

- $(4, 5, 1)_{RNS(7|6|5)}$. Note that, $x_1 = 4$, $m_1 = 7$, $x_2 = 5$, $m_2 = 6$, $x_3 = 1$, $m_3 = 5$. It can be seen that $x_1 + x_3$ is odd and x_1 is not greater than x_2, thus we employ Equation (5.33) as follows:

$$\begin{aligned} X &= 6(5-4) + 5 + 42 \left| \frac{5}{2} + \frac{5}{2} - 5 \right|_5 \\ &= 6 + 5 + 42|0|_5 \\ &= 11, \end{aligned}$$

  again as it should.

5.3 Hardware Implementation

The hardware structure of the proposed scheme, depicted in Figure 5.3 is based on the equations in Theorems 5.4 and 5.5. In adder A, residue x_1 is subtracted from x_2 and next the result is multiplied by m_2. The 3-input adder B computes $(\frac{x_1+x_3}{2} - x_2)$ and can be implemented as a 3:2 Carry Save Adder (CSA) followed by a Carry Propagate Adder (CPA). We prove later in this section that only one corrective addition or subtraction is required to compute the modulo-m_3 operation and this can be combined and done in the same step with the possible addition of $\frac{m_3}{2}$ term. We also demonstrate that only one corrective addition is required to compute the final modulo-M and it can be also embedded with the modulo-m_3. The above mentioned operations are implemented by adder C with a selectable input. The hardware implementation removes the fractions by shifting left all the operands involved in adder B and C, thereby extending the two adders with one bit. Finally, the output of adder C, without the rightmost bit to account for the previous shift, is multiplied by the multiplier m_1m_2 and the result is summed together by adder D with the one from the multiplier m_2. The extra input x_2 for adder D can be embedded in the m_2 multiplier according to the principle of merged arithmetic, thus D can be actually implemented as a standard 2:1 adder.

Next, we demonstrate that no explicit m_3 modulo operation is required by $\left|\frac{(x_1+x_3)}{2} - x_2\right|_{m_3}$ and $\left|\frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2\right|_{m_3}$ by analyzing the four possible extreme cases as follows:

Case 1: $(x_1+x_3) = 0$ and $x_2 = 2n-1$. This results in the most negative value one may get. In this case the modulus in Equation (5.24) reduces to $|-x_2|_{m_3}$. To perform the modulo m_3 operation we need to do corrective additions. Given that $m_3 + (-x_2) = (2n-1) - (2n-1) = 2n - 1 - 2n + 1 = 0$, for any positive integer n, only one corrective addition with m_3 is required to compute the modulo.

Case 2: $(x_1 + x_3)$ is even and has the maximum possible value and x_2 is zero. This is the largest positive value one may get and the modulus in Equation (5.24) reduces to $|\frac{(x_1+x_3)}{2}|_{m_3}$. Given that $m_3 - \frac{(x_1+x_3)}{2} = (2n-1) - \frac{(2n+2n-2)}{2} = 2n - 1 - 2n + 1 = 0$ the maximum sum in the modulo adder cannot exceed m_3, thus one subtraction with m_3 is required.

Case 3: $(x_1 + x_3) = 1$ and $x_2 = 2n - 1$. In this case the modulus in Equation (5.33) reduces to $|\frac{m_3}{2} + \frac{1}{2} - x_2|_{m_3}$. Given that in this case $\frac{m_3}{2} + \frac{1}{2} - x_2$ is always negative and that

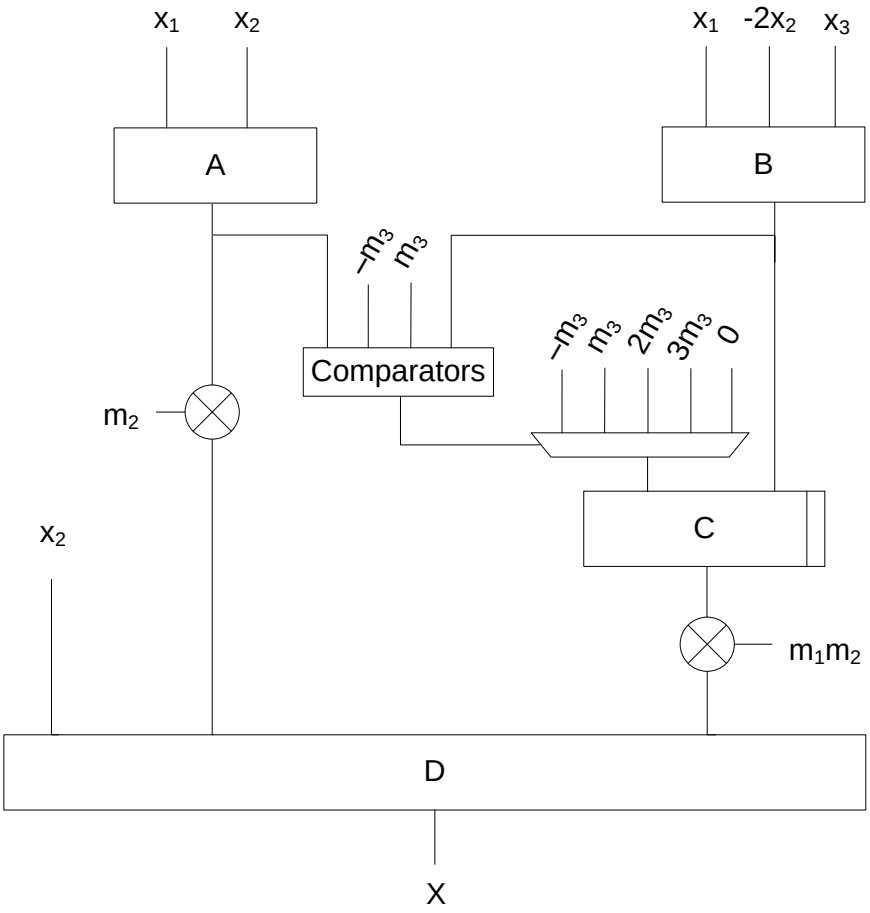


Figure 5.1: Hardware Structure of Our Proposal

$m_3 + \frac{m_3}{2} + \frac{1}{2} - x_2 = 2n - 1 + n - \frac{1}{2} + \frac{1}{2} - 2n + 1 = n > 0$, for any integer n, only one corrective addition with m_3 is required to compute the modulo.

Case 4: $(x_1 + x_3)$ odd has the maximum possible value and x_2 is zero. The modulus in Equation (5.33) reduces to $|\frac{m_3}{2} + \frac{(x_1+x_3)}{2}|_{m_3}$. Given that $2m_3 - (\frac{m_3}{2} + \frac{(x_1+x_3)}{2}) = 2(2n-1) - (\frac{2n-1}{2} + \frac{(2n+2n-2)-1}{2}) = 4n - 2 - 3n + 2 = n > 0$, for any positive integer n, one corrective subtraction of m_3 is required to compute the modulo.

This means that the modulo m_3 operation can be implemented with at most one

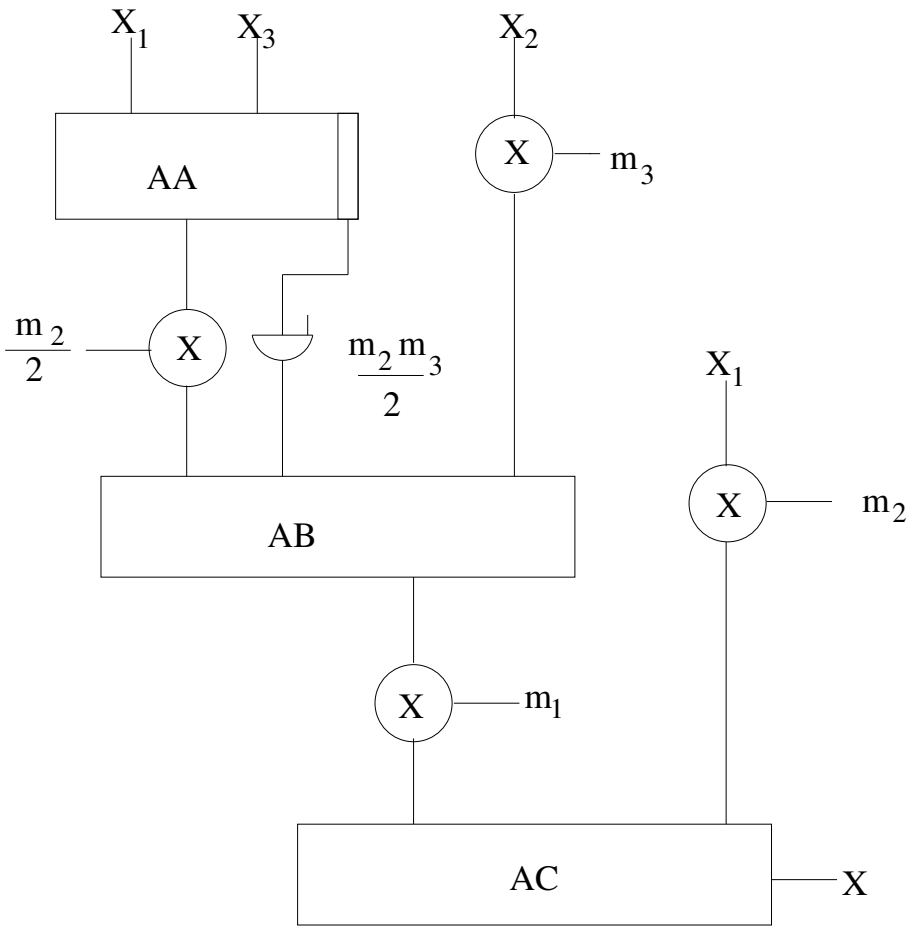


Figure 5.2: Converter Data Path for GC.

corrective addition or subtraction. In the following, we prove that the addition of the $\frac{m_3}{2}$ term can be actually postponed and embedded into the correction step required for the modulo-m_3 operation without any delay overhead. For that, we remove $\frac{m_3}{2}$ as adder C input and revisit the 4 correction cases analyzed above.

If (x_1+x_3) is even, the term $\frac{m_3}{2}$ is not part of the calculation and the correction can be done as usual. If $(x_1 + x_3)$ is odd, the tentative sum at the output of adder B is $\frac{(x_1+x_3)}{2} - x_2$ instead of $\frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2$, thus it is smaller with $\frac{m_3}{2}$ than it should actually be. Taking that into consideration, the correction rules change to:

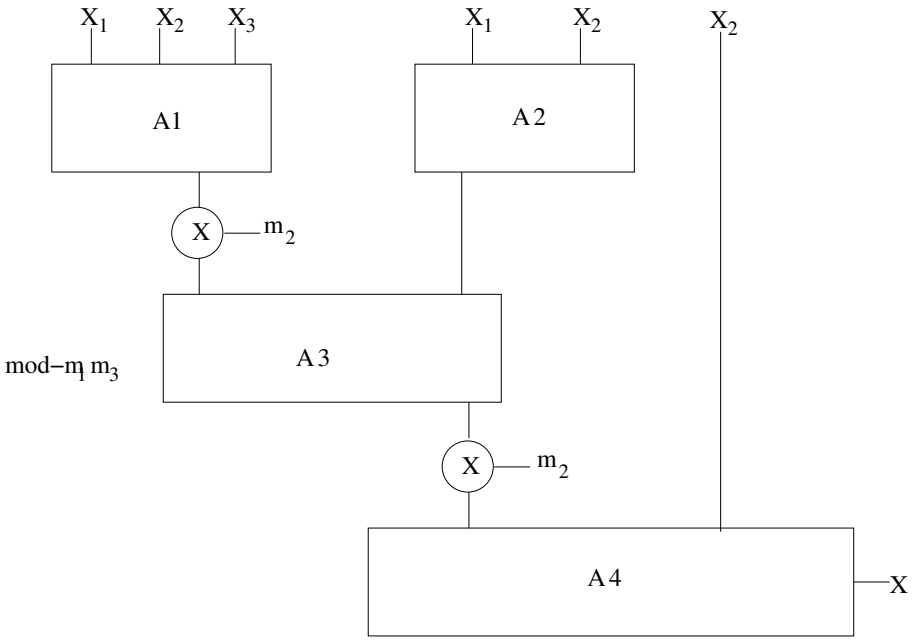


Figure 5.3: Converter Data Path for SAW.

1. $(x_1 + x_3)$ even:
   - if tentative sum is smaller than 0 add m_3;
   - if tentative sum is larger than or equal to m_3 subtract m_3;
   - otherwise do nothing;
2. $(x_1 + x_3)$ odd:
   - if tentative sum is smaller than $-\frac{m_3}{2}$ add $\frac{3m_3}{2}$;
   - if tentative sum is greater than or equal to $\frac{m_3}{2}$ subtract $\frac{m_3}{2}$;
   - otherwise add $\frac{m_3}{2}$.

As indicated by Equations (5.31) and (5.39), the final modulo-M also does not require explicit implementation. The scheme is simplified by moving this addition before the $m_1 m_2$ multiplication, hence transforming it into a corrective m_3 addition. As mentioned in the previous section, this correction must be applied when both of the following statements hold true:

$$x_1 > x_2 \tag{5.40}$$

$$\begin{cases} \left|\frac{(x_1+x_3)}{2} - x_2\right|_{m_3} = 0 & (x_3+x_1) \text{ even} \\ \left|\frac{m_3}{2} + \frac{(x_1+x_3)}{2} - x_2\right|_{m_3} = 0 & (x_3+x_1) \text{ odd} \end{cases} \tag{5.41}$$

We now combine the modulo-m_3 operations by revisiting the correction rules:

1. $(x_1 + x_3)$ even:
   - if tentative sum is smaller than 0 add m_3;
     Equation (5.41) holds true when the tentative sum is equal to $-m_3$, but we can see from *Case 1* that Equation (5.40) does not hold true, hence the extra m_3 addition is not needed;
   - if tentative sum is zero and Equation (5.40) is true (the sign bit of adder A is 1) add m_3;
   - otherwise do nothing;
     Equation (5.41) holds true when the tentative sum is equal to m_3 and we can see from *Case 2* that this happens when x_2 is zero and $x_1 = 2n$, hence Equation (5.40) also holds true. But the required m_3 addition cancels out the previous m_3 corrective subtraction for the modular-m_3 adder;
2. $(x_1 + x_3)$ odd:
   - if tentative sum is smaller than $-\frac{m_3}{2}$ add $\frac{3m_3}{2}$;
     From *Case 3*, it can be seen that the minimum value for the tentative sum is $\frac{1}{2} - x_2 < -\frac{3m_3}{2}$, so Equation (5.41) does not hold true. Thus, no extra m_3 addition is required;
   - if tentative sum is larger than $\frac{m_3}{2}$ subtract $\frac{m_3}{2}$;
     Equation (5.41) holds true when the tentative sum is equal to $\frac{m_3}{2}$. Following this, $x_1+x_3-2x_2 = 2m_3 \Rightarrow x_1-x_2 = 2m_3-x_3+x_2$. Since $2m_3 - x_3 > 0$ Equation (5.40) also holds true, thus the correction becomes $-\frac{m_3}{2} + m_3 = \frac{m_3}{2}$;
   - otherwise add $\frac{m_3}{2}$.

In this way all modulo operations have been replaced by a single corrective addition or subtraction, which in turn greatly reduces the complexity of the converter.

5.4 Performance Analysis

It is well known that the direct implementation of Equation (5.1) is not effective because it requires large magnitude multipliers and also a modulo M operation, (where M is a large number), is required before the decimal number can be obtained. If the New CRT in [129] is utilized, the complexity is somehow reduced as smaller numbers are used and the modulo operation is m_1m_3 instead of M. It is quite clear that the reverse converter proposed in this chapter is processing smaller numbers in the multiplication when compared to the ones in Equations (5.1), (5.2), (5.3), (5.14), and (5.15) because the involved modulo operation is m_3.

Metrics	SAW	GC	Our proposal
Area	4 adders 2 multipliers	3 adders 4 multipliers	4 adders 2 multipliers
Delay	3/4 additions 2 multiplications	3 additions 2 multiplications	3/4 additions 1 multiplications 1 comparator + 1 MUX
Modulo	m_1m_3	m_2m_3	m_3

Table 5.1: Performance Comparison

The converter in [2] for the moduli set under consideration has been shown to outperform the one in [99]. Recently, the converter proposed in [41] was also shown to be better than the one in [2] through a detailed critical path analysis, which takes the hardware implementation details into consideration. In view of that, we compare our proposal with the equivalent converters in [41] and [2], further referred in the rest of this chapter by GC and SAW, respectively. The number of arithmetic operations required for our proposal - Equations (5.24), and (5.33), the one in [2] - Equation (5.4), and the one in [41]-Equations (5.14) and (5.15)- are summarized in Table 5.1. As one can observe in the table, the proposed converter requires less delay. Moreover, due to the fact that our scheme operates on smaller magnitude operands, it results in less complex adders and multipliers, which potentially results in even faster and smaller implementations.

Figures 5.3, 5.2, and 5.3 depict the hardware realization of the proposed converter, the converters GC and SAW, respectively. On the critical path, our proposal requires three 2:1 adders and one multiplier, SAW requires one 4:1 adder (even though not included in the figure, the term $2z_0n$ has to be part of the A2

adder inputs), two 2:1 adders, and two multipliers while GC requires one 3:1 adder, two 2:1 adders, and two multipliers. Consequently, the new converter introduced here is less complex and it is faster than state of the art equivalent converters. Additionally, we also carried out experimental comparison by describing the proposed converter, the converter in [41] and the one in [2] in VHDL and implementing them on Xilinx Spartan 3 xa3s200-4-ftg256 FPGA, with Xilinx ISE 10.1.03. In order to properly evaluate the relations between

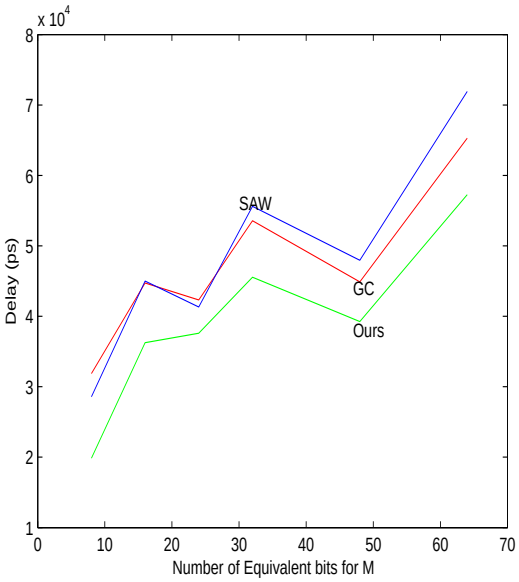


Figure 5.4: Delay Improvement

these converters, several reverse converters were implemented for a wide range of values of n, up to an equivalent dynamic range of 64 bits. The converter performances evaluated in terms of area (expressed as number of FPGA slices), and delay (in ns obtained by post-place route static timing analysis) is depicted in Figures 5.4 and 5.5, respectively.

As expected, since all the three converters have somehow similar architectures, all of them present roughly the same area and delay profile. The downward spikes that all the converters present in both area and delay graphs occur when n is a power of 2. For this special values of n the logic synthesizer optimizes the design by replacing the multipliers with simple shifters, thereby greatly reducing both the occupied area and the time of conversion. Apart from these

spikes, we can observe some non-monotonic area and delay variations which are also induced by logic optimizations made by the synthesizer, for certain n values.

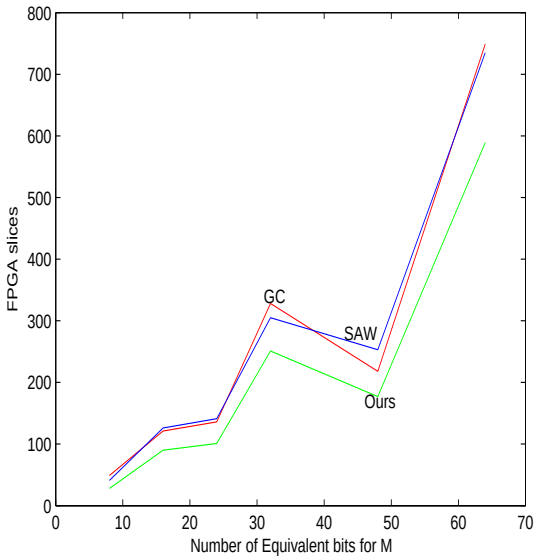


Figure 5.5: Area Improvement

Table 5.2: Synthesized Results: Area Comparison

M	n	Our Area (slices)	GC Area (slices)	SAW Area (slices)
2^8	4	28	49	41
2^{16}	21	90	121	126
2^{24}	129	101	136	141
2^{32}	813	251	328	305
2^{48}	32769	177	218	253
2^{64}	1321123	589	749	735

The obtained values suggest that, on average, the proposed converter is capable of performing the reverse conversion 15.8% and 14% faster with 21% and 27% area decrease, when compared to the reverse converters GC and SAW, respectively. Exact values of number of occupied FPGA slices, conversion time, and average power consumption are presented in Tables 5.2, 5.3, and 5.4,

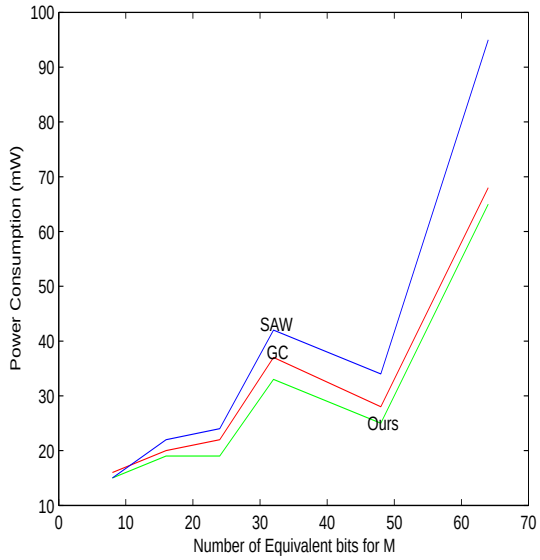


Figure 5.6: Power savings

Table 5.3: Synthesized Results: Delay Comparison

M	n	Our Delay (ps)	GC Delay (ps)	SAW Delay (ps)
2^8	4	19857	31865	28575
2^{16}	21	36262	44718	44999
2^{24}	129	37599	42329	41312
2^{32}	813	45543	53556	55609
2^{48}	32769	39256	44896	47955
2^{64}	1321123	57261	65295	71930

for some common dynamic ranges. For this particular cases, our converter consumes on average 8% and 18% less power when compared with SAW and GC, respectively.

5.5 Conclusion

In this chapter, we proposed a new reverse converter for the moduli set $\{2n+1, 2n, 2n-1\}$. First, we simplified the traditional CRT in order to

Table 5.4: Synthesized Results: Power-Comparison

M	n	Proposed Converter	GC Converter	SAW Converter
2^8	4	15	16	15
2^{16}	21	19	20	22
2^{24}	129	19	22	24
2^{32}	813	33	37	42
2^{48}	32769	25	28	34
2^{64}	1321123	65	68	95

obtain a reverse converter that utilizes mod-$(2n-1)$ operations instead of mod-$(2n)(2n-1)$ and $(2n+1)(2n-1)$ operations required by the converters in [41] and [2], respectively. Next, we transformed the needed corrective addition of M into a corrective m_3 addition. We then presented a novel low complexity implementation that does not require the explicit use of modulo operation in the conversion process.

The performance of the proposed converter is evaluated both theoretically, in terms of the required number of arithmetic operations and experimentally by FPGA implementation. Theoretically speaking, the proposed converter requires lesser number of arithmetic operations when compared to the state of the art equivalent converters.

For the experimental assessment, the converters were described in VHDL and implemented on Xilinx Spartan 3 FPGA. We used a wide range of values of n for the implementation. The synthesis results indicate that, on average, the proposed converter outperforms the state of the art with about 14% and 21%, and 8% in terms of conversion time, area cost, and power consumption, respectively.

Given that powers of two moduli sets offer larger dynamic range when compared to the powers of two related moduli set discussed in this chapter, we investigate powers of two moduli set in the next chapter.

Chapter 6

A Converter for the Moduli Set $\{2^{n+1}-1, 2^n, 2^n-1\}$

Powers of two moduli sets are very useful in building adder based reverse converters. They are also more applicable to Digital Signal Processing (DSP) applications requiring larger dynamic range when compared to the power of two related moduli set discussed in the previous chapter. Due to this fact, we focuss on a power of two moduli set in this chapter. Many algorithms have been designed for performing the reverse conversion with different choices of moduli sets, e.g., $\{2^n, 2^n-1, 2^n+1\}$ [133], $\{2^n, 2^n-1, 2^{n-1}-1\}$ [57], $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ [86], $\{2^n, 2^{n+1}-1, 2^n-1\}$ [75,85], $\{2n+1, 2n-1, 2n\}$ [99], $\{2n+2, 2n+1, 2n\}$ [44,100]. Recently, the moduli set $\{2^n, 2^{n+1}-1, 2^n-1\}$ was proposed in [85] by removing the modulus (2^n+1) from the 4-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ proposed in [121]. This is due to the fact that performing the modulo (2^n+1)-type arithmetic is complex and degrades the entire RNS system performance in terms of both area and delay. Three reverse converters have been proposed in [85] for the moduli set $\{2^n, 2^{n+1}-1, 2^n-1\}$. Two of those converters are very efficient delay wise at the expense of very high area cost while the third one has a reduced area cost at the expense of very low conversion speed. In [75], an area efficient residue to binary converter was proposed with a slower conversion speed when compared to the two speed efficient converters of [85]. In this chapter, two novel memoryless residue to binary converters for the $\{2^{n+1}-1, 2^n, 2^n-1\}$ moduli set are proposed.

The remainder of this chapter is organized as follows. Section 6.1 presents state of the art equivalent converters. In Section 6.2, a novel memoryless re-

verse converter is proposed. We resolve the dynamic range limitation problem in Section 6.3. Section 6.4 describes the hardware implementation of the proposed algorithm. Section 6.5 evaluates the performance of the proposed algorithm, while the chapter is concluded in Section 6.6.

6.1 Background

For a moduli set $\{m_1, m_2, m_3, ..., m_k\}$ with the dynamic range $M = \prod_{i=1}^{k} m_i$, the residue number $(x_1, x_2, x_3, ..., x_k)$ can be converted into the decimal number X, according to the Chinese Reminder Theorem (CRT), as follows [111]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M , \tag{6.1}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

For the moduli set under investigation, efficient converters have been presented in [85] and [75]. Three converters were presented in [85]. One of those converters, which is area efficient is based on the traditional Mixed Radix Conversion (MRC) while the remaining two, which are based on the traditional CRT are speed efficient. The converter presented in [75], which has been shown to be more effective when compared to the ones in [85] is based on Equation (6.2) and is represented by:

$$X = x_1 + 2^n Y, \tag{6.2}$$

where

$$\begin{aligned} Y &= \left| (-2^{n+1}+3)(x_2 - x_1) \right. \\ &\quad \left. +(2^{n+1}-1)(x_3 - x_1) \right|_{(2^{n+1}-1)(2^n-1)} , \end{aligned} \tag{6.3}$$

which was further simplified as

$$\begin{aligned} Y &= |(2^n-1)(2x_1 - 4x_2 + 2x_3) \\ &\quad +(x_3 - x_1)|_{(2^{n+1}-1)(2^n-1)} . \end{aligned} \tag{6.4}$$

In the following section, we present a new effective reverse converter for the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$ by first simplifying Equation (6.1).

6.2 Modulo-$(2^{n+1}-1)$ Reverse Conversion

Given the RNS number (x_1, x_2, x_3) for the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2^{n+1}-1, 2^n, 2^n-1\}$, the proposed algorithm computes the decimal equivalent of this RNS number based on a further simplification of the well-known traditional CRT. First, we show that the computation of the multiplicative inverses can be eliminated for this moduli set resulting into a memoryless reverse converter. Next, we obtain a reverse converter that only requires modulo-$(2^{n+1}-1)$ operations. We further reduce the hardware complexity in order to obtain a pure adder based RNS-to-binary converter.

Theorem 6.1 *Given the moduli set $\{m_1, m_2, m_3\}$ with $m_1 = 2^{n+1}-1$, $m_2 = 2^n$, $m_3 = 2^n - 1$, the following hold true:*

$$|(m_1 m_2)^{-1}|_{m_3} = 1, \tag{6.5}$$
$$|(m_1 m_3)^{-1}|_{m_2} = 1, \tag{6.6}$$
$$|(m_2 m_3)^{-1}|_{m_1} = -4. \tag{6.7}$$

Proof:
If it can be demonstrated that $|1 \times (m_1 m_2)|_{m_3} = 1$, then 1 is the multiplicative inverse of $(m_1 m_2)$ with respect to m_3. $|1 \times (m_1 m_2)|_{m_3}$ is given by: $\left|2^n(2^{n+1}-1)\right|_{2^n-1} = \left|2^{2n+1} - 2^n\right|_{2^n-1} = \left|\left|2^{2n+1}\right|_{2^n-1} - |2^n|_{2^n-1}\right|_{2^n-1} = |2-1|_{2^n-1} = 1$, thus (6.5) holds true.

In the same way if $|1 \times (m_1 m_3)|_{m_2} = 1$, then 1 is the multiplicative inverse of $(m_1 m_3)$ with respect to m_2. $|1 \times (m_1 m_3)|_{m_2}$ is given by: $\left|(2^{n+1}-1)(2^n-1)\right|_{2^n} = \left|2^{2n+1} - 2^{n+1} - 2^n + 1\right|_{2^n} = \left|2^{2n+1} - 3(2^n) + 1\right|_{2^n} = \left|\left|2^{2n+1}\right|_{2^n} - |3(2^n)|_{2^n} + |1|_{2^n}\right|_{2^n} = |0-0+1|_{2^n} = 1$, thus (6.6) holds true.

Finally, if $|(-4) \times (m_2 m_3)|_{m_1} = 1$, then -4 is the multiplicative inverse of $(m_2 m_3)$ with respect to m_1. $|(-4) \times (m_2 m_3)|_{m_1}$ is given by: $|(-4)(2^n)(2^n-1)|_{2^{n+1}-1} = \left|(-2^2)(2^{2n} - 2^n)\right|_{2^{n+1}-1} = \left|-2^{2n+2} + 2^{n+2}\right|_{2^{n+1}-1} = \left|\left|-2^{2n+2}\right|_{2^{n+1}-1} + \left|2^{n+2}\right|_{2^{n+1}-1}\right|_{2^{n+1}-1} = |-1+2|_{2^{n+1}-1} = |1|_{2^{n+1}-1} = 1$, thus (6.7) holds true.

The following relations introduce necessary preliminary information, which enable a further simplification of the traditional CRT in the subsequent theorem. Given the moduli set $\{m_1, m_2, m_3\}$ with $m_1 = 2^{n+1}-1$, $m_2 = 2^n$,

$m_3 = 2^n - 1$, the following hold true:

$$m_1 = 2m_2 - 1, \tag{6.8}$$
$$m_1 = 2m_3 + 1, \tag{6.9}$$
$$m_3 = m_2 - 1. \tag{6.10}$$

The following theorem introduces a simplified way to compute the decimal equivalent of the RNS number.

Theorem 6.2 *The decimal equivalent of the residues $(x_{i,i=1,3})$ for the moduli set $\{m_{i,i=1,3}\}$ in the form $\{2^{n+1}-1, 2^n, 2^n-1\}$, assuming $X \in [0, M - m_3^2)$, can be computed as follows:*

$$X = m_2 \left\lfloor \frac{X}{m_2} \right\rfloor + x_2 \tag{6.11}$$

where

$$\left\lfloor \frac{X}{m_2} \right\rfloor = x_3 - x_2 + m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \tag{6.12}$$

Proof:
Since Equation (6.11) follows the basic integer division definition in RNS, which is always true, we only need to show the correctness of Equation (6.12). For $k = 3$, Equation (6.1) becomes:

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M \tag{6.13}$$

By substituting Equations (6.5), (6.6), and (6.7) into Equation (6.13) we obtain the following:

$$X = \left| m_2 m_3 (-4) x_1 + m_1 m_3 x_2 + m_1 m_2 x_3 \right|_M ,$$

and by substituting Equations (6.8) and (6.9) in the above equation, we obtain:

$$\begin{aligned} X &= \left| -4m_2 m_3 x_1 + m_3(2m_2 - 1)x_2 + m_2(2m_3 + 1)x_3 \right|_M , \\ &= \left| -4m_2 m_3 x_1 + 2m_2 m_3 x_2 - m_3 x_2 + 2m_2 m_3 x_3 + m_2 x_3 \right|_M \end{aligned} \tag{6.14}$$

Equation (6.14) can be further simplified by using the following lemma presented in [131]:

$$\left| a m_1 \right|_{m_1 m_2} = m_1 \left| a \right|_{m_2} \tag{6.15}$$

Applying Equation (6.15), Equation (6.14) becomes:

$$X = \left| m_2x_3 - m_3x_2 + m_2m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \right|_M \quad (6.16)$$

If Equation (6.10) is utilized in Equation (6.16), we have:

$$\begin{aligned} X &= \left| m_2x_3 - x_2(m_2-1) + m_2m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \right|_M \\ &= \left| m_2x_3 - m_2x_2 + x_2 + m_2m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \right|_M \end{aligned}$$

Dividing both sides of the above equation by m_2 and taking the floor, we shall have:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = \left| x_3 - x_2 + m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \right|_{m_1m_3} \quad (6.17)$$

From Equation (6.12), it can be seen easily that Equation (6.17) is the same as $\left| \left\lfloor \frac{X}{m_2} \right\rfloor \right|_{m_1m_3}$. The next stage of the proof is to demonstrate that the corrective addition required for the calculation of the mod-m_1m_3 can be avoided in most of the cases. By definition of modulus, we have:

$$\begin{aligned} 0 &\leq \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \leq m_1 - 1 \\ 0 &\leq m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \leq m_1m_3 - m_3 \end{aligned} \quad (6.18)$$

Observing that $0 \leq x_3 < m_3$ and $0 \leq x_2 < m_2\,(= m_3 + 1)$,
and using Equation (6.18), we have:

$$\begin{aligned} -m_3 &\leq -x_2 \leq x_3 - x_2 + m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} \\ &< m_1m_3 - m_3 + m_3\,(= m_1m_3) \end{aligned} \quad (6.19)$$

Thus, one corrective addition of m_1m_3 is required in order to obtain the correct result when $x_3 - x_2 + m_3 \left| -4x_1 + 2x_2 + 2x_3 \right|_{m_1} < 0$.

Further, we show that if we slightly restrict the RNS dynamic range, no corrective addition is required. For the numbers that require corrective addition, the following condition holds true:

$$\begin{aligned} -x_2 + m_1m_3 &\leq \left\lfloor \frac{X}{m_2} \right\rfloor < m_1m_3 \\ M - m_2x_2 &\leq m_2 \left\lfloor \frac{X}{m_2} \right\rfloor < M \\ M - (m_2-1)x_2 &\leq X < M \\ M - m_3m_3 &\leq X < M. \end{aligned} \quad (6.20)$$

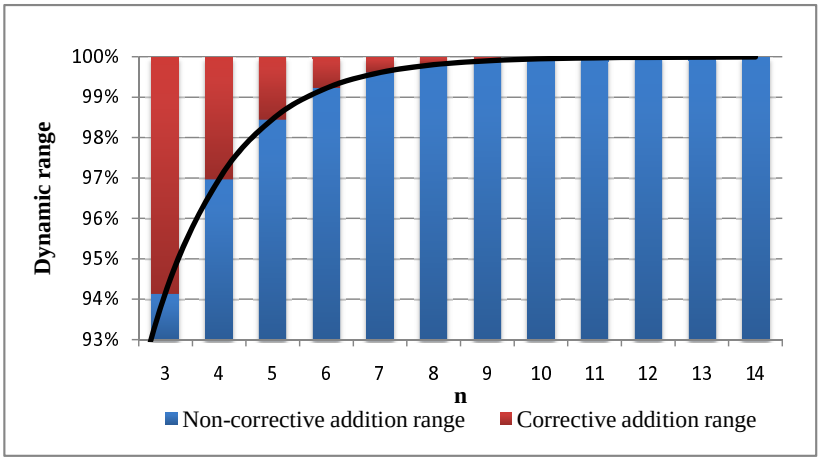


Figure 6.1: Dynamic range Vs n

Therefore, the numbers within the interval $[0, M-(m_3)^2)$ require no corrective addition and thus, Equation (6.12) holds true.

Thus, the numbers that need corrective additions lay in the interval $[M-(m_3)^2, M-1]$, which is on the top part of the dynamic range. The ratio that this range amounts from the dynamic range given by n is plotted in Figure 6.1. It can be observed that the region requiring corrective addition exponentially decreases as n increases. For reference, Table 6.1 provides the exact values for the dynamic and limited range for n up to 8. Generally speaking, if the numbers in the interval $[M-(m_3)^2, M-1]$ require corrective addition, the numbers within the interval $[0, M-(m_3)^2)$ require no corrective addition and thus, Equation (6.12) holds true.

Table 6.1: Limited and Non-limited Dynamic Range

n	2	3	4	5	6	7	8
m_1	7	15	31	63	127	255	511
m_2	4	8	16	32	64	128	256
m_3	3	7	15	31	63	127	255
M	84	840	7440	62496	512064	4145280	33358080
M_p	72	784	7200	61504	508032	4129024	33292800

The hardware complexity of Equation (6.12) can be further reduced by using the following properties from [86]:

Property 1: Modulo (2^s-1) multiplication of a residue number by 2^t, where s and t are positive integers, is equivalent to t-bit circular left shifting.

Property 2: Modulo (2^s-1) of a negative number is equivalent to the one's complement of the number, which is obtained by subtracting the number from (2^s-1).

Suppose that Equation (6.12) is represented by:

$$\begin{aligned}\left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^n-1)A \\ &= x_3 - x_2 + 2^n A - A, \end{aligned} \tag{6.21}$$

where

$$A = |u_1 + u_2 + u_3|_{2^{n+1}-1}\,. \tag{6.22}$$

For simplicity sake, let us represent Equation (6.21) by the following:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_1 + B_2 + B_3, \tag{6.23}$$

where

$$\begin{aligned} B_1 &= -x_2, \\ B_2 &= 2^n A + x_3, \\ B_3 &= -A. \end{aligned} \tag{6.24}$$

Let the binary representations of the residues be the following:

$$\begin{aligned} x_1 &= (x_{1,n}x_{1,n-1}...x_{1,1}x_{1,0}), \\ x_2 &= (x_{2,n-1}x_{2,n-2}...x_{2,1}x_{2,0}), \\ x_3 &= (x_{3,n-1}x_{3,n-2}...x_{3,1}x_{3,0}). \end{aligned}$$

In Equation (6.22), u_1, u_2, and u_3 are represented as follows:

$$\begin{aligned} u_1 &= \left|-2^2 x_1\right|_{2^{n+1}-1} \\ &= \left|-2^2 (x_{1,n}x_{1,n-1}...x_{1,0})\right|_{2^{n+1}-1} \\ &= -(\underbrace{x_{1,n-2}x_{1,n-3}\cdots x_{1,0}x_{1,n}x_{1,n-1}}_{n+1}) \\ &= (\underbrace{\overline{x}_{1,n-2}\overline{x}_{1,n-3}\cdots \overline{x}_{1,0}\overline{x}_{1,n}\overline{x}_{1,n-1}}_{n+1}), \end{aligned}$$

$$\begin{aligned} u_2 &= |2x_2|_{2^{n+1}-1} \\ &= (\underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}0}_{n+1}), \end{aligned}$$

and

$$\begin{aligned} u_3 &= |2x_3|_{2^{n+1}-1} \\ &= (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}0}_{n+1}). \end{aligned}$$

Given that A has the following binary representation:

$$A = (\underbrace{a_n a_{n-1}\cdots a_1 a_0}_{n+1}),$$

then B_2 will be given by

$$\begin{aligned} B_2 &= \underbrace{(\underbrace{a_n a_{n-1}\cdots a_1 a_0}_{n+1}\underbrace{00\cdots 0}_{n}),}_{2n+1} \\ &\quad +(\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\ &= (\underbrace{a_n a_{n-1}\cdots a_0 x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{2n+1}) \end{aligned} \tag{6.25}$$

In Equation (6.23), in order to carry out the summation, B_1 and B_3 must have equal number of bits (i.e., $(2n+1)$-bits) as B_2. They are represented as:

$$\begin{aligned} B_1 &= -x_2 \\ &= -\underbrace{(\underbrace{000\cdots0}_{n+1}\underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}}_{n})}_{2n+1} \\ &= \underbrace{(\underbrace{111\cdots11}_{n+1}\underbrace{\overline{x}_{2,n-1}\overline{x}_{2,n-2}\cdots\overline{x}_{2,0}}_{n})}_{2n+1}, \end{aligned} \tag{6.26}$$

$$\begin{aligned} B_3 &= -A \\ &= -\underbrace{(\underbrace{000\cdots0}_{n}\underbrace{a_n a_{n-1}\cdots a_0}_{n+1})}_{2n+1} \\ &= \underbrace{(\underbrace{111\cdots11}_{n}\underbrace{\overline{a}_n\overline{a}_{n-1}\cdots\overline{a}_0}_{n+1})}_{2n+1}. \end{aligned} \tag{6.27}$$

6.3 Dynamic Range Limitation Problem

In this section, we resolve the dynamic range limitation problem. In Equation (6.17), if $x_3 - x_2 + m_3|-4x_1+2x_2+2x_3|_{m_1} < 0$, then $|-4x_1+2x_2+2x_3|_{m_1} = 0$ since $m_3 \geq x_2$.
Thus, we have this negative value iff:

$$x_2 > x_3 \wedge |-4x_1+2x_2+2x_3|_{m_1} = 0 \tag{6.28}$$

For this case of condition Equation (6.28), only one corrective addition of m_1m_3 is required as shown in the previous section. Therefore, Equation (6.17) can be written as:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = x_3 - x_2 + m_1m_3 \tag{6.29}$$

Equation (6.29) can be simplified as follows:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^{n+1}-1)(2^n-1) \\ &= x_3 - x_2 + 2^{2n+1} - 2^{n+2} + 2^n + 1 \end{aligned} \tag{6.30}$$

By using the following notations:

$$\begin{aligned}
B_4 &= -x_2 \\
&= \underbrace{(\underbrace{111\cdots11}_{n+2}\underbrace{\overline{x}_{2,n-1}\overline{x}_{2,n-2}\cdots\overline{x}_{2,0}}_{n})}_{2n+2}, \\
B_5 &= 1-2^{n+2} \\
&= (\underbrace{100\cdots00}_{n+3}), \\
B_6 &= 2^n + x_3 \\
&= (\underbrace{100\cdots00}_{n+1}) + (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\
&= (\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}), \\
B_7 &= B_5 + B_6 \\
&= (\underbrace{101x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+3}), \\
B_8 &= B_7 + 2^{2n+1} \\
&= (\underbrace{100\cdots00}_{n-1})(\underbrace{101x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+3})
\end{aligned} \tag{6.31}$$

Equation (6.30) may be simplified as follows:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_4 + B_8 \tag{6.32}$$

6.4 Hardware Implementation

The hardware implementations of the proposed reverse converter, which does not cover the entire dynamic range, namely CI is based on Equations (6.22) and (6.23). In Figure 6.2, u_1, u_2, and u_3 are added by CSA1 with end around carry (EAC) producing s_1 and c_1. Next, these must be added modulo $2^{n+1}-1$ in order to obtain A. To speed up this addition, we utilize anticipated computation. We compute s_1 + c_1 for both $cin = 0$ and $cin = 1$ and we select the right result with a MUX. B_2 is easily obtained by concatenating the operand x_3 with the result of n-bit left shift of A. This concatenation does not require any hardware resources. The three operands B_1, B_2, and B_3 are added using

CSA2 with EAC. It should be noted that in order to make B_1 and B_3 $(2n+1)$-bit numbers, 1's are appended to the result of complementations, as given in Equations (6.26) and (6.27). Thus, the most significant $(n+1)$-bits from CSA2 are reduced to half adders (HAs). Moreover, since n most significant bits of CSA2 all have two inputs equal to 1, the final one's complement adder will always generate an end-around carry. Taking this into consideration, the one's complement adder can be reduced to a normal CPA3 with a constant carry-in equals to 1. The final result, which is computed based on Equation (6.11) is obtained just by a shift and a concatenation operation with no computational hardware.

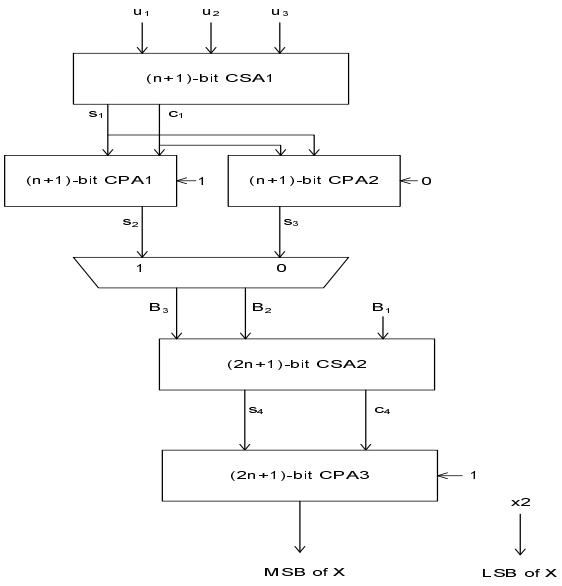


Figure 6.2: Proposed Converter CI

Given that in reality, the numbers that fall outside the range $[0, M-(m_3)^2)$ may be of interest, we resolve the dynamic range limitation problem and propose a second converter namely CII based on Equations (6.22) , (6.23) and (6.32). The hardware implementations of CII, which is valid for the entire dynamic range $[0, M-1]$, is depicted in Figure 6.3.

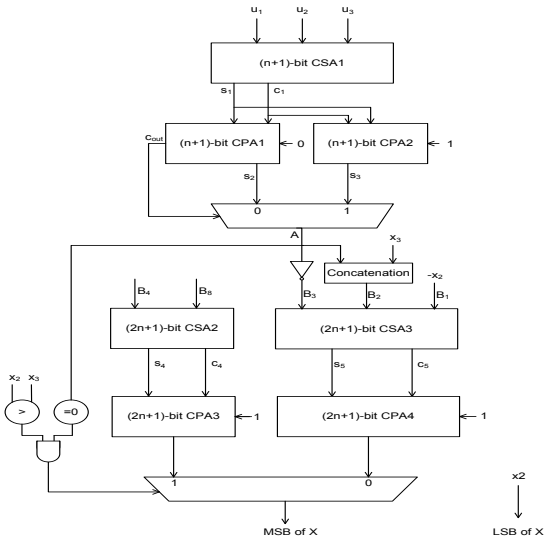


Figure 6.3: Proposed Converter CII

6.5 Performance Evaluation

The performance of the proposed residue to binary converters is evaluated in terms of hardware cost and conversion delay. In order to evaluate the performance of the proposed converters, we compare them with the best state of the art equivalent converters proposed in [75]. The result of this comparison is presented in Table 6.2. In the table, we have the converters CI and CII as the

Table 6.2: Area-Delay Comparisons

Converters	FA	HA	Delay
CIII	$5n+3$	$2n+1$	$(4n+6)t_{FA}+t_{MUX}$
CI	$5n$	$n+4$	$2nt_{FA}+(n+2)t_{HA}+t_{MUX}$
CII	$6n-1$	$n+5$	$2nt_{FA}+(n+2)t_{HA}+2t_{MUX}$

converters proposed in this chapter and CIII, the one in [75]. The theoretical results presented in Table 6.2 indicate that the proposed converter CI outperforms CIII in all terms. Converter CII, which is valid for the entire dynamic range, maintains almost the same lower delay than CIII at an additional hard-

ware cost.

Table 6.3: Synthesized Results: Area-Delay Comparison

n	3	4	5	8
CI Area	22	27	34	57
CII Area	30	35	44	74
CIII Area	44	43	55	94
CI Delay (ns)	18.401	21.276	25.621	33.604
CII Delay (ns)	19.482	21.429	25.273	34.111
CIII Delay (ns)	22.143	22.331	24.126	34.419
CI Area-Delay	404.822	574.452	871.114	1915.428
CII Area-Delay	584.46	750.015	1112.012	2524.214
CIII Area-Delay	974.292	960.233	1326.930	3235.386

We also carried out an experimental assessment by implementing the proposed converters and CIII using Xilinx ISE 10.1 software on a Xa3s200-4vqg100 FPGA. The synthesis results are given in terms of the number of slices and input-to-output gate delays (in nano seconds). Table 6.3 presents the results for various dynamic range requirements (different values of n). Contrary to the theoretical analysis, the results indicate that, on average, the proposed converter CI reduces the area by about 42% when compared with the current most effective CIII converter, with a small improvement in the speed of conversion. However, the proposed full-range converter CII is about 29.48% smaller, still with some speed improvement over CIII, but slower than the one achieved by CI.

6.6 Conclusion

In this chapter, we proposed two new novel memoryless residue to binary converters for the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$. First, we simplified the CRT to obtain a reverse converter that requires mod-$(2^{n+1}-1)$ instead of both of mod-$(2^{n+1}-1)$ and mod-(2^n-1) required by state of the art converter. Second, we further reduced the resulting architecture in order to obtain a reverse converter that utilizes only CSAs and CPAs. We reduced the large $(2^{n+1}-1)(2^n-1)$ adder to a simple adder with the consequence of a reduction in the dynamic range from M to M-$(2^n-1)^2$. We demonstrated that the reduction penalty decreases exponentially as n increases and for values of $n \geq 5$, the reduction penalty is negligible ($\leq 0.0078\%$). We resolved the dynamic

range limitation problem and proposed another converter, which is valid for the entire dynamic range. Theoretically speaking, the two proposed converters are faster than the most effective equivalent state of the art converter, but one of them requires more hardware resources. Finally, we implemented our proposals and the most effective equivalent state of the art converter on a Xilinx Spartan 3 FPGA. The obtained results show that, on average and contrary to the theoretical results, our schemes significantly reduced the area cost with no delay penalty.

Given that the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$, presented in this chapter is insufficient for application requiring larger dynamic range, the next chapter presents moduli sets with larger dynamic range.

Chapter 7

Larger Dynamic Range type $(2^n + 1)$-free Moduli Set

Moduli set choice is an important issue since the complexity and the speed of the resulting conversion algorithm depend on the chosen moduli set. Several structures have been proposed to perform the reverse conversion for different moduli sets, e.g., $\{2^n, 2^n - 1, 2^n + 1\}$ [27], $\{2^{n+1} + 1, 2^{n+1} - 1, 2^n\}$ [86], $\{2^n, 2^{n+1} - 1, 2^n - 1\}$ [75, 85], $\{2n + 2, 2n + 1, 2n\}$ [44]. As mentioned in Chapter 6 the moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$ was proposed in [85] by the elimination of the modulus $(2^n + 1)$ from the 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ proposed in [121]. The motivation for this is related to the fact that the modulo $(2^n + 1)$-type arithmetic is more complex and degrades the entire RNS performance both in terms of area cost and conversion delay. However, the moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$, which utilizes only fast modulo operations, is insufficient for applications requiring larger dynamic range. Consequently, the moduli set $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ was proposed in [88] together with a reverse converter based on Mixed Radix Conversion (MRC). This chapter presents two novel high speed reverse converters for the $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ moduli set.

The rest of this chapter is organized as follows. In Section 7.1, the new limited dynamic range algorithm for reverse conversion is proposed. We resolve the domain restriction problem in Section 7.2. Section 7.3 presents the hardware implementation of the proposed reverse converters, while Section 7.4 gives a performance comparison with the best known equivalent state of the art converter. Section 7.5 concludes this chapter.

7.1 A Speed Efficient Reverse Converter

For a moduli set $\{m_1, m_2, m_3, ..., m_k\}$, the residues $(x_1, x_2, x_3, ..., x_k)$ can be converted into the corresponding decimal number X, according to the Chinese Remainder Theorem (CRT), as follows [131]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M , \tag{7.1}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = \frac{M}{m_i}$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

The complexity of Equation (7.1) is greatly reduced by using the moduli set $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$. Consequently, we present an algorithm that computes the decimal equivalent of the Residue Number System (RNS) integer (x_1, x_2, x_3) based on a further simplification of Equation (7.1). First, we show that the computation of the multiplicative inverses can be eliminated resulting into a memoryless reverse converter.

Theorem 7.1 *Given the moduli set* $\{m_1, m_2, m_3\}$ *with* $m_1 = 2^{2n+1} - 1, m_2 = 2^n, m_3 = 2^n - 1$*, the following hold true:*

$$|(m_1 m_2)^{-1}|_{m_3} = 1, \tag{7.2}$$

$$|(m_1 m_3)^{-1}|_{m_2} = 1, \tag{7.3}$$

$$|(m_2 m_3)^{-1}|_{m_1} = 2^{2n+1} - 2^{n+2} - 3. \tag{7.4}$$

Proof: If it can be demonstrated that $|1 \times (m_1 m_2)|_{m_3} = 1$, then 1 is the multiplicative inverse of $(m_1 m_2)$ with respect to m_3:

$$\begin{aligned} R_1 &= \left| 2^n (2^{2n+1} - 1) \right|_{2^n - 1} \\ &= \left| \left| 2^{3n+1} \right|_{2^n - 1} - |2^n|_{2^n - 1} \right|_{2^n - 1} \\ &= |2 - 1|_{2^n - 1} = 1, \end{aligned}$$

thus Equation (7.2) holds true.

In the same way, if $|1 \times (m_1 m_3)|_{m_2} = 1$, then 1 is the multiplicative inverse of $(m_1 m_3)$ with respect to m_2:

$$\begin{aligned} R_2 &= \left| (2^{2n+1} - 1)(2^n - 1) \right|_{2^n} \\ &= \left| 2^{3n+1} - 2^{2n+1} - 2^n + 1 \right|_{2^n} \\ &= \left| \left| 2^{3n+1} \right|_{2^n} - \left| 2^{2n+1} \right|_{2^n} - |2^n|_{2^n} + |1|_{2^n} \right|_{2^n} \\ &= |0 - 0 - 0 + 1|_{2^n} = 1, \end{aligned}$$

thus Equation (7.3) holds true.

Again, if $\left|(2^{2n+1}-2^{n+2}-3)\times(m_2m_3)\right|_{m_1}=1$, then $(2^{2n+1}-2^{n+2}-3)$ is the multiplicative inverse of (m_2m_3) with respect to m_1:

$$\begin{aligned}
R_3 &= \left|(2^{2n+1}-2^{n+2}-3)(2^n)(2^n-1)\right|_{2^{2n+1}-1}\\
&= \left|(2^{2n+1}-2^{n+2}-3)(2^{2n}-2^n)\right|_{2^{2n+1}-1}\\
&= \left|2^{4n+1}-3(2^{2n})+2^{2n+2}-2^{3n+2}-2^{3n+1}+3(2^n)\right|_{2^{2n+1}-1}\\
&= \left|2^{4n+1}-3(2^{2n})+2(2^{2n+1})-6(2^{3n})+3(2^n)\right|_{2^{2n+1}-1}\\
&= \left|2^{4n+1}-3(2^{2n})+2(2^{2n+1})-3(2^n)(2^{2n+1}-1)\right|_{2^{2n+1}-1}\\
&= \Big|\left|2^{4n+1}\right|_{2^{2n+1}-1}-\left|3(2^{2n})\right|_{2^{2n+1}-1}+\left|2(2^{2n+1})\right|_{2^{2n+1}-1}\\
&\quad -\left|3(2^n)(2^{2n+1}-1)\right|_{2^{2n+1}-1}\Big|_{2^{2n+1}-1}\\
&= \left|\frac{1}{2}-\frac{3}{2}+2+0\right|_{2^{2n+1}-1}=1,
\end{aligned}$$

thus Equation (7.4) holds true.

The following important relations will be utilized in the subsequent theorem: Given the moduli set $\{m_1,m_2,m_3\}$ with $m_1=2^{2n+1}-1, m_2=2^n, m_3=2^n-1$, the following holds true:

$$m_1 = 2m_2m_2-1, \tag{7.5}$$

$$m_2 = m_3+1. \tag{7.6}$$

The following theorem introduces a simplified way to compute the decimal equivalent of the RNS number (x_1,x_2,x_3).

Theorem 7.2 *The decimal equivalent of the residues (x_1,x_2,x_3) for the moduli set $\{m_1,m_2,m_3\}$ in the form $\{2^{2n+1}-1,2^n,2^n-1\}$, assuming $X\in[0,M-m_3^2)$, can be computed as follows:*

$$X=m_2\left\lfloor\frac{X}{m_2}\right\rfloor+x_2 \tag{7.7}$$

where

$$\begin{aligned}
\left\lfloor\frac{X}{m_2}\right\rfloor &= x_3-x_2+m_3\left|(-2^{n+2}-2)x_1\right.\\
&\quad \left.+2m_2x_2+2m_2x_3+2x_3\right|_{m_1}
\end{aligned} \tag{7.8}$$

Proof: Since Equation (7.7) follows the basic integer division definition in RNS, which is always true, we only need to show the correctness of Equation (7.8). For $k = 3$, (7.1) becomes:

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M \tag{7.9}$$

Substituting Equations (7.2), (7.3), and (7.4) into Equation (7.9) we obtain the following:

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

and by substituting Equations (7.5) and (7.6) in the above equation, we obtain:

$$\begin{aligned} X &= \big|(m_2 m_3)(2^{2n+1} - 2^{n+2} - 3) x_1 \\ &\quad + m_3(2m_2 m_2 - 1)x_2 + m_2(2m_2 m_2 - 1)x_3|_M , \\ X &= \big|(m_2 m_3)(2^{2n+1} - 2^{n+2} - 3) x_1 + 2m_2 m_2 m_3 x_2 - m_3 x_2 \\ &\quad + 2m_2 m_2 (m_3 + 1)x_3 - m_2 x_3|_M . \end{aligned} \tag{7.10}$$

Equation (7.10) can be further simplified by using the following lemma, presented in [131]:

$$|a m_1|_{m_1 m_2} = m_1 |a|_{m_2} \tag{7.11}$$

Applying Equations (7.11) and (7.6), (7.10) becomes:

$$\begin{aligned} X &= \big| m_2 x_3 - m_3 x_2 + m_2 m_3 \big| (2^{2n+1} - 2^{n+2} - 3) x_1 \\ &\quad + 2m_2 x_2 + 2m_2 x_3 + 2x_3|_{m_1} \big|_M \end{aligned} \tag{7.12}$$

If Equation (7.6) is applied to simplify even further in Equation (7.12), we have:

$$\begin{aligned} X &= \big| m_2 x_3 - x_2(m_2 - 1) + m_2 m_3 \big| (2^{2n+1} - 2^{n+2} - 3) x_1 \\ &\quad + 2m_2 x_2 + 2m_2 x_3 + 2x_3|_{m_1} \big|_M \\ &= \big| m_2 x_3 - m_2 x_2 + x_2 + m_2 m_3 \big| (2^{2n+1} - 2^{n+2} - 3) x_1 + \\ &\quad 2m_2 x_2 + 2m_2 x_3 + 2x_3|_{m_1} \big|_M \end{aligned}$$

Dividing both sides of the above equation by m_2 and taking the floor, we have:

$$\begin{aligned}
\left\lfloor \frac{X}{m_2} \right\rfloor &= \big|x_3 - x_2 + m_3 \big| (2^{2n+1} - 2^{n+2} - 3)x_1 \\
&\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \\
&= \Big|x_3 - x_2 + m_3 \Big| |2^{2n+1}x_1|_{m_1} - 2^{n+2}x_1 \\
&\quad -3x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \\
&= \big|x_3 - x_2 + m_3 \big| x_1 - 2^{n+2}x_1 - 3x_1 \\
&\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \\
&= \big|x_3 - x_2 + m_3 \big| -2^{n+2}x_1 - 2x_1 \\
&\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \qquad (7.13)
\end{aligned}$$

From Equation (7.8), it can be seen easily that Equation (7.13) is the same as $\left|\left\lfloor \frac{X}{m_2} \right\rfloor\right|_{m_1m_3}$. The next stage of the proof is to demonstrate that the corrective addition required for the calculation of the mod-m_1m_3 can be avoided in most of the cases. By definition of modulus, we have:

$$\begin{aligned}
0 &\leq \big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \leq m_1 - 1 \\
0 &\leq m_3 \big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \\
&\leq m_1m_3 - m_3 \qquad (7.14)
\end{aligned}$$

Observing that $0 \leq x_3 < m_3$ and $0 \leq x_2 < m_2\,(= m_3 + 1)$,
and using (7.14), we have:

$$\begin{aligned}
-m_3 &\leq -x_2 \leq x_3 - x_2 \\
&+ \quad m_3 \big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \\
&< m_1m_3 - m_3 + m_3\,(= m_1m_3) \qquad (7.15)
\end{aligned}$$

Thus, one corrective addition of m_1m_3 is required in order to obtain the correct result when $x_3 - x_2 + m_3 \big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} < 0$.

Further, we show that if we slightly restrict the RNS dynamic range, no corrective addition is required. For the numbers that require corrective addition,

the following condition holds true:

$$\begin{aligned} -x_2 + m_1 m_3 &\leq \left\lfloor \frac{X}{m_2} \right\rfloor < m_1 m_3 \\ M - m_2 x_2 &\leq m_2 \left\lfloor \frac{X}{m_2} \right\rfloor < M \\ M - (m_2 - 1)x_2 &\leq X < M \\ M - m_3 m_3 &\leq X < M. \end{aligned} \tag{7.16}$$

Therefore, the numbers within the interval $[0, M-(m_3)^2)$ require no corrective addition and thus, Equation (7.8) holds true.

Thus, the numbers that need corrective additions lay in the interval $[M - (m_3)^2, M - 1]$, which is on the top part of the dynamic range. Figure 7.1 depicts the corrective and non-corrective addition regions. It can be observed that the corrective addition range exponentially decreases as n increases. Generally speaking, if the numbers in the interval $[M - (m_3)^2, M - 1]$ require corrective addition, the numbers within the interval $[0, M - (m_3)^2]$ require no corrective addition and thus, Equation (7.8) holds true.

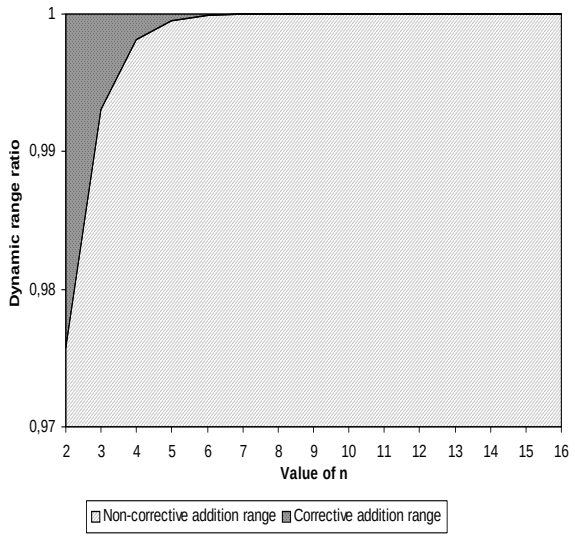


Figure 7.1: Dynamic range ratio Vs n-values

We can further reduce the hardware complexity of the reverse converter by simplifying Equation (7.8) using the following two properties [86].

Property 1: Modulo $(2^s - 1)$ multiplication of a residue number by 2^t, where s and t are positive integers, is equivalent to t bit circular left shifting.

Property 2: Modulo $(2^s - 1)$ of a negative number is equivalent to the one's complement of the number, which is obtained by subtracting the number from $(2^s - 1)$.

Suppose that Equation (7.8) is written as:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^n - 1)A \\ &= x_3 - x_2 + 2^n A - A, \end{aligned} \tag{7.17}$$

where

$$A = |u_0 + u_1 + u_2 + u_3 + u_4|_{2^{2n+1}-1}\,. \tag{7.18}$$

For simplicity sake, let us represent Equation (7.17) as:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_1 + B_2 + B_3, \tag{7.19}$$

where

$$\begin{aligned} B_1 &= -x_2, \\ B_2 &= 2^n A + x_3, \\ B_3 &= -A. \end{aligned} \tag{7.20}$$

Let the binary representations of the residues be:

$$\begin{aligned} x_1 &= (x_{1,2n} x_{1,2n-1} ... x_{1,1} x_{1,0}), \\ x_2 &= (x_{2,n-1} x_{2,n-2} ... x_{2,1} x_{2,0}), \\ x_3 &= (x_{3,n-1} x_{3,n-2} ... x_{3,1} x_{3,0}). \end{aligned}$$

In Equation (7.18), u_0, u_1, u_2, u_3, and u_4 are represented as follows:

$$\begin{aligned}
u_0 &= \left|-2^{n+2}x_1\right|_{2^{2n+1}-1} \\
&= \left|-2^{n+2}(x_{1,2n}x_{1,2n-1}...x_{1,0})\right|_{2^{2n+1}-1} \\
&= \left|-2^{n+2}(\underbrace{x_{1,2n}x_{1,2n-1}...x_{1,n-1}}_{n+2}\underbrace{x_{1,n-2}...x_{1,0}}_{n-1})\right|_{2^{2n+1}-1} \\
&= -(\underbrace{\cdots x_{1,0}x_{1,2n}x_{1,2n-1}\cdots}_{2n+1}) \\
&= (\underbrace{\overline{x}_{1,n-2}\overline{x}_{1,n-3}...\overline{x}_{1,1},\overline{x}_{1,0}}_{n-1}\underbrace{\overline{x}_{1,2n}\overline{x}_{1,2n-1}...\overline{x}_{1,n-1}}_{n+2}),
\end{aligned}$$

$$\begin{aligned}
u_1 &= |-2x_1|_{2^{2n+1}-1} \\
&= |-2(x_{1,2n}x_{1,2n-1}...x_{1,0})|_{2^{2n+1}-1} \\
&= -(\underbrace{x_{1,2n-1}\cdots x_{1,0}x_{1,2n}}_{2n+1}) \\
&= (\underbrace{\overline{x}_{1,2n-1}\cdots\overline{x}_{1,0}\overline{x}_{1,2n}}_{2n+1}),
\end{aligned}$$

$$\begin{aligned}
u_{i,i=2,3} &= \left|2^{n+1}x_i\right|_{2^{2n+1}-1} \\
&= \left|2^{n+1}(x_{i,n}x_{i,n-1}...x_{i,0})\right|_{2^{2n+1}-1} \\
&= (\underbrace{x_{i,n}x_{i,n-1}\cdots x_{i,0}}_{n}\underbrace{00\cdots0}_{n+1}),
\end{aligned}$$

$$\begin{aligned}
u_4 &= |2x_3|_{2^{2n+1}-1} \\
&= |2(x_{3,n}x_{3,n-1}...x_{3,0})|_{2^{2n+1}-1} \\
&= (\underbrace{00\cdots0}_{n}\underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}}_{n}\underbrace{0}_{1}).
\end{aligned}$$

Given the binary representation:

$$A = (\underbrace{a_{2n}a_{2n-1}\cdots a_1a_0}_{2n+1}),$$

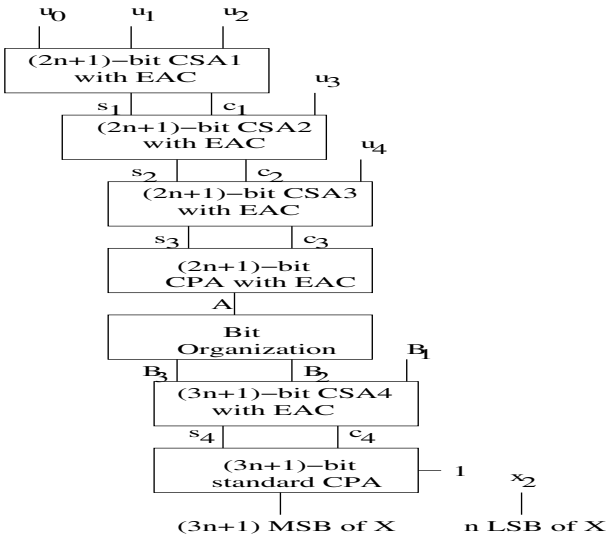


Figure 7.2: Proposal-I

B_2 will be written as:

$$\begin{aligned} B_2 &= (\underbrace{a_{2n}a_{2n-1}\cdots a_1a_0}_{2n+1}\underbrace{00\cdots0}_{n}) \\ &\quad +(\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\ &= \underbrace{(a_{2n}a_{2n-1}\cdots a_0x_{3,n-1}x_{3,n-2}\cdots x_{3,0})}_{3n+1} \end{aligned} \tag{7.21}$$

In (7.19), in order to carry out the summation, B_1 and B_3 must have equal

number of bits, i.e., $(3n + 1)$-bits, as B_2. They are represented as:

$$\begin{aligned} B_1 &= -x_2 \\ &= -(\underbrace{000\cdots0}_{2n+1}\underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}}_{n}) \\ &= (\underbrace{111\cdots11}_{2n+1}\underbrace{\overline{x}_{2,n-1}\overline{x}_{2,n-2}\cdots\overline{x}_{2,0}}_{n}), \qquad (7.22) \\ B_3 &= -A \\ &= -(\underbrace{000\cdots0}_{n}\underbrace{a_n a_{n-1}\cdots a_0}_{2n+1}) \\ &= (\underbrace{111\cdots11}_{n}\underbrace{\overline{a}_n\overline{a}_{n-1}\cdots\overline{a}_0}_{2n+1}). \qquad (7.23) \end{aligned}$$

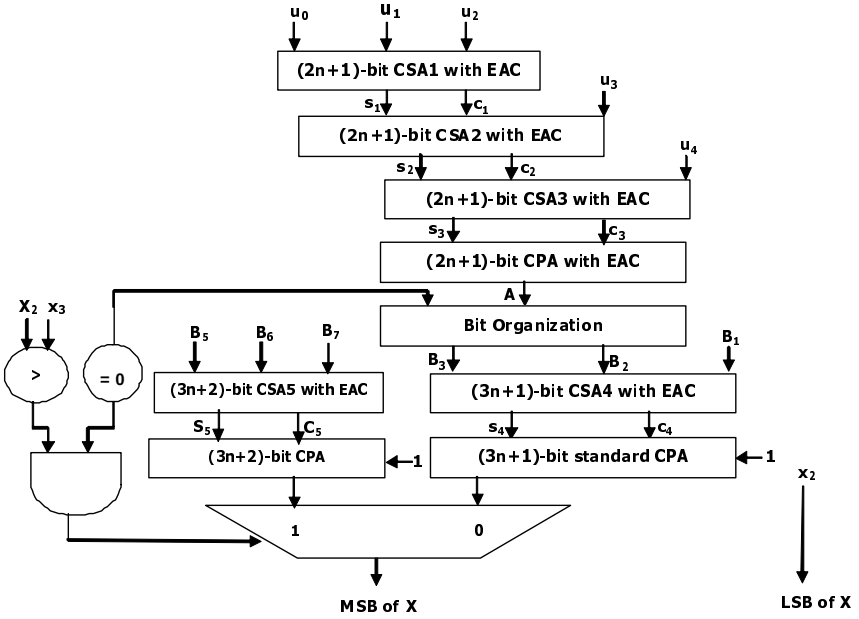


Figure 7.3: Proposal-II

7.2 Domain Restriction Problem

In this section, we resolve the dynamic range limitation problem for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set. In Equation (7.13), if $x_3 - x_2 + m_3 \left|(-2^{n+2}-2)x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\right|_{m_1} < 0$, then $\left|(-2^{n+2}-2)x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\right|_{m_1} = 0$ since $m_3 \geq x_2$.
Thus, we have this negative result iff:

$$x_2 > x_3 \wedge \left|(-2^{n+2}-2)x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\right|_{m_1} = 0 \tag{7.24}$$

For this case of condition (7.24), only one corrective addition of m_1m_3 is required as shown in the previous section. Therefore, Equation (7.13) can be written as:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = x_3 - x_2 + m_1m_3 \tag{7.25}$$

Equation (7.25) can be simplified as follows:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^{2n+1}-1)(2^n-1) \\ &= x_3 - x_2 + 2^{3n+1} - 2^{2n+1} - 2^n + 1 \\ &= x_3 - x_2 + 2^{3n+1} - 2^{2n+1} + 2^n \end{aligned} \tag{7.26}$$

Equation (7.26) may be simplified as follows:

$$\begin{aligned}
B_5 &= -x_2 \\
&= \underbrace{(\underbrace{111\cdots11}_{2n+2}\underbrace{\overline{x}_{2,n-1}\overline{x}_{2,n-2}\cdots\overline{x}_{2,0}}_{n})}_{3n+2}, && (7.27)\\
B_4 &= 2^n + x_3 \\
&= (\underbrace{100\cdots00}_{n+1}) + (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\
&= (\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}), && (7.28)\\
B_6 &= B_4 + 2^{3n+1} \\
&= (\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}) + (\underbrace{100\cdots0}_{3n+2}) \\
&= (\underbrace{100\cdots0}_{2n+1})(\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}), && (7.29)\\
B_7 &= -2^{2n+1} \\
&= -(\underbrace{100\cdots00}_{2n+2}) \\
&= -(\underbrace{00\cdots0}_{n})(\underbrace{100\cdots00}_{2n+2}) \\
&= (\underbrace{11\cdots1}_{n})(\underbrace{011\cdots1}_{2n+2}). && (7.30)
\end{aligned}$$

Therefore,

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_5 + B_6 + B_7 \qquad (7.31)$$

7.3 Hardware Implementation

The hardware structure of the proposed reverse converter, which does not cover the entire dynamic range, namely proposal-I is based on Equations (7.18) and (7.19). In Figure 7.2, u_0, u_1, u_2, u_3, and u_4 are added by Carry Save Adders (CSAs) with end-around carries (EACs) producing the values s_3 and c_3. These values must be added modulo $2^{2n+1}-1$ in order to obtain A, i.e., with a one's complement adder, namely a Carry Propagate Adder (CPA) with EAC. B_2 is easily obtained by concatenating the operand x_3 with the n-bit left

Table 7.1: Area and Delay Comparisons

Components	MRC Converter	Proposal-I	Proposal-II
FA	$9n+2$	$9n+2$	$13n+6$
HA	2	$5n+4$	$7n+4$
XNOR	$2n$	$-$	$-$
OR	$2n$	$-$	$-$
Delay	$(10n+5)t_{FA}$	$(7n+7)t_{FA}$	$(7n+9)t_{FA}$

shift of A. This concatenation does not require any additional hardware. The three operands B_1, B_2, and B_3 are added using a CSA with EAC. It should be noted that in order to make B_1 and B_3 $(3n+1)$-bit numbers, 1's are appended to the result of complementations, as given in Equations (7.22) and (7.23). Thus, the addition of the most significant $(2n+1)$-bits performed in this CSA can be performed by Half Adders (HAs). Moreover, since the n MSB of the CSA all have two inputs equal to 1, the final one's complement adder will always generate an end-around carry. Taking this into consideration the one's complement adder can be reduced to a normal CPA with a constant carry-in equals to 1. The final result, computed from Equation (7.7) is obtained simply by a shift and a concatenation operation not requiring any additional hardware resources.

Given that the numbers that fall outside the range $[0, M-m_3^2)$ may be of interest, we resolve the domain restriction problem and propose a second converter namely proposal-II, which is based on Equations (7.18), (7.19) and (7.31). The hardware implementations of proposal-II, which is valid for the entire dynamic range $[0, M-1]$, is depicted in Figure 7.3.

7.4 Performance Evaluation

In order to evaluate the performance of the proposed converters, we compare them with the best known state of the art equivalent reverse converter proposed in [88]. This comparison has been carried out by both performing a theoretical analysis and experimentally by implementing on an FPGA. The theoretical analysis is presented in Table 8.1. This table indicates that, in terms of area, the reverse converter in [88] is slightly better than the herein proposed converters. Proposal-I requires $(5n+4)$HA, proposal-II requires more $(4n+4)$FA with $(7n+4)$HA against 2HA, $2n$ XNOR gates, and $2n$ OR gates required by the one in [88]. However, in terms of delay, proposal-I and proposal-II are

$(3n - 2)t_{FA}$ and $(3n - 4)t_{FA}$, respectively, faster than the reverse converter in [88].

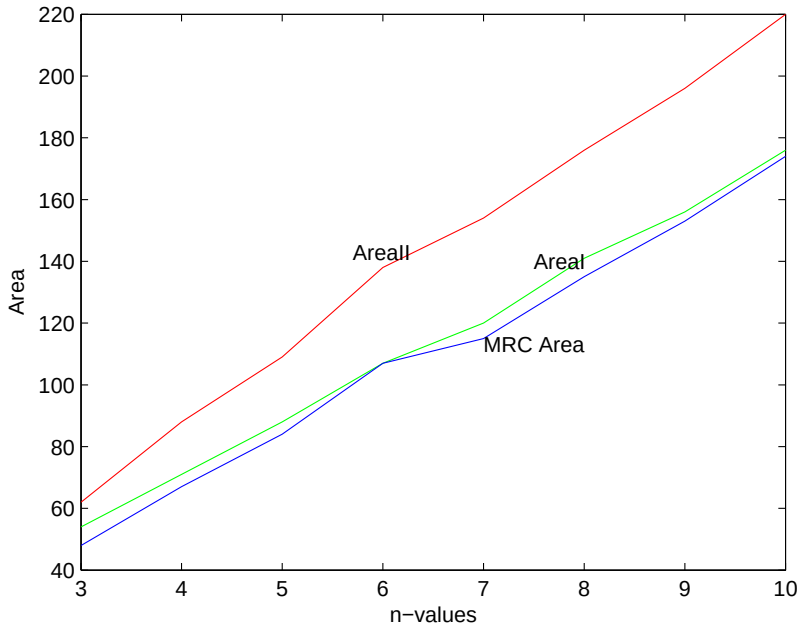


Figure 7.4: Area Comparison

Table 7.2: Implementation Results

n	AreaI (FPGA Slices)	AreaII (FPGA Slices)	MRC Area (FPGA Slices)	DelayI (ns)	DelayII (ns)	MRC Delay (ns)
3	54	62	48	30.396	30.757	36.901
4	71	88	67	34.048	32.185	34.418
5	88	109	84	38.582	38.582	41.668
6	107	138	107	37.129	37.278	45.600
7	120	154	115	41.198	40.447	47.318
8	141	176	135	40.319	40.694	48.908
9	156	196	153	45.253	45.436	51.628
10	176	220	174	45.012	47.256	53.757

For the experimental assessment, the converters were described in VHDL and

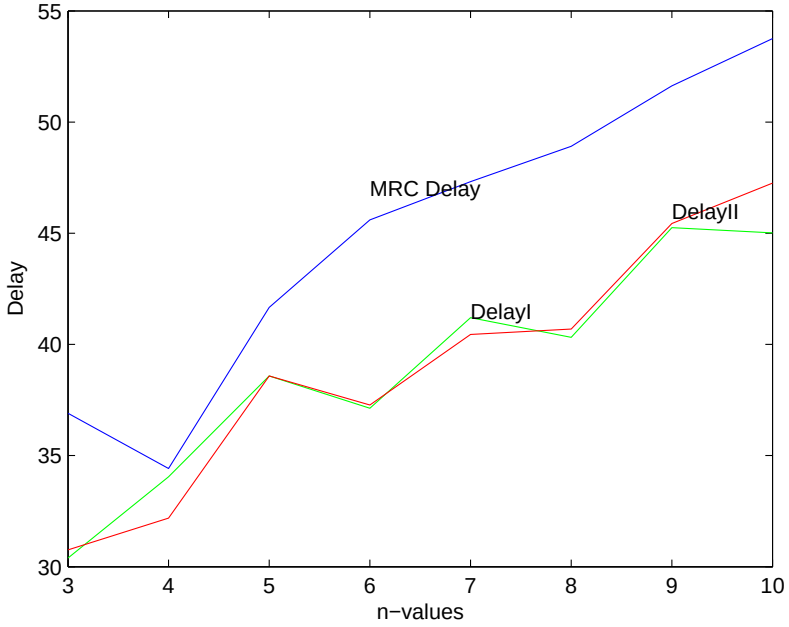


Figure 7.5: Conversion Speed Comparison

Table 7.3: AT^2 Comparison

n	AT^2-I	AT^2-II	MRC-AT^2
3	49892	58652	65361
4	82308	91157	79368
5	130994	162254	145843
6	147506	191772	222492
7	203673	251938	257484
8	229213	291456	322919
9	319462	404628	407814
10	356590	491289	502828

we implemented them on FPGA. In order to properly evaluate the relation between these reverse converters, several reverse converters were implemented for a wide range of values of n. Experimental results are presented in Tables 7.2 and 7.3. In comparison with the reverse converter in [88], the obtained

values suggest that, on average, proposal-I and proposal-II, respectively, are capable of performing the reverse conversion 13.40% and 13.20% faster, with extra costs of 3.29% and 22.75% in terms of area. To adequately compare the conversion structures, the Area-Time Square (AT^2) efficiency metric was used. This metric suggests that proposal-I and proposal-II, respectively, are 24% and 3% more efficient than the one in [88]. Figures 7.4 and 7.5 present the performance of the reverse converters, respectively, in terms of area and delay.

7.5 Conclusion

In this chapter, two novel high speed memoryless residue to binary converters for $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ moduli set are proposed. First, we simplified the traditional CRT to obtain a reverse conveter that requires mod-$(2^{n+1} - 1)$ instead of both of mod-$(2^{2n+1} - 1)$ and mod-$(2^n - 1)$ required by the reverse converter in [88]. We further simplified the resulting architecture in order to obtain a pure adder based memoryless converter, which is made up of only CSAs and CPAs. This leads to a sturcture that is amenable to efficient VLSI chip realization. We resolved the domain restriction problem and propose another reverse converter, which is valid for the entire dynamic range.

The performance of the proposed reverse converters are evaluated both theoretically, in terms of gate analysis, and for an FPGA implementation. The theoretical analysis indicates that the two proposed reverse converters are faster than the one in [88] in terms of delay, but one of them requires more hardware resources. For the experimental comparison, we described the proposed reverse converters and the one in [88] in VHDL and carried out the implementation on an FPGA. We used a wide range of values of n for the implementation. The synthesis results indicate that the obtained values suggest that, on average, proposal-I and proposal-II, respectively, are capable of performing the reverse conversion 13.40% and 13.20% faster, with extra costs of 3.29% and 22.75% in terms of area when compared with the one in [88]. The AT^2 metric suggests that proposal-I and proposal-II, respectively, are 24% and 3% more efficient than the one in [88].

In the next chapter, we present efficient reverse converters for the moduli set $\{2^{2n+1} - 1, 2^{2n}, 2^n - 1\}$ since the moduli set, which is investigated in this chapter is underutilizing the binary channel.

Chapter 8

Binary Channel Enhancement Conversion Techniques

As discussed in the previous chapter, multiplication by powers of 2 with respect to the $(2^n + 1)$ modulus is not as simple as left circular rotation in a $(2^n - 1)$ modulus. Due to this reason, efficient reverse converter was proposed for the moduli set $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ in the previous chapter. The disadvantage of this moduli set is that the binary channel is underutilized. In this chapter, we present three new reverse converters for the $\{2^{2n+1} - 1, 2^{2n}, 2^n - 1\}$ moduli set by extending the modulus 2^n in the $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ moduli set. We propose two Chinese Remainder Theorem (CRT) based converters, one of which covers the entire dynamic range while the second one does not, and a Mixed Radix Conversion (MRC) based converter, which is valid for the entire range.

The remainder of this chapter is organized as follows. In Section 8.1, the new limited dynamic range CRT based algorithm for reverse conversion is proposed. We resolve the domain restriction problem in Section 8.2. Section 8.3 presents the MRC based conversion technique. Section 8.4 presents the hardware implementation of both of the CRT and the MRC based conversion techniques, while Section 8.5 gives a performance comparison with the similar best known state of the art converters. The chaper is concluded in Section 8.6.

8.1 Limited Dynamic Range CRT Technique

Given the Residue Number System (RNS) integer (x_1, x_2, x_3) for the moduli $\left\{2^{2n+1}-1, 2^{2n}, 2^n-1\right\}$, the proposed algorithm computes the decimal equivalent of this RNS number based on a further simplification of the well-known traditional CRT. First, we wish to show that the moduli $2^{2n+1}-1$, 2^{2n}, and 2^n-1 are relatively prime.

Theorem 8.1 *The moduli set* $\left\{2^{2n+1}-1, 2^{2n}, 2^n-1\right\}$ *contains pairwise relatively prime moduli.*

Proof: From Euclidean theorem, we have:
$gcd(a,b) = gcd(b, |a|_b)$,
therefore,

$$\begin{aligned} gcd(2^{2n+1}-1, 2^{2n}) &= gcd(2^{2n}, \left|2^{2n+1}-1\right|_{2^{2n}}) \\ &= gcd(2^{2n}, 1) = 1 \end{aligned}$$

Also,

$$\begin{aligned} gcd(2^{2n+1}-1, 2^n-1) &= gcd(2^n-1, \left|2^{2n+1}-1\right|_{2^n-1}) \\ &= gcd(2^n-1, 1) = 1 \end{aligned}$$

Again

$$\begin{aligned} gcd(2^{2n}, 2^n-1) &= gcd(2^n-1, \left|2^{2n}\right|_{2^n-1}) \\ &= gcd(2^n-1, 1) = 1 \end{aligned}$$

Thus, the moduli $2^{2n+1}-1$, 2^{2n}, and 2^n-1 are relatively prime since all the greatest common divisors are equal to 1.

Theorem 8.2 *Given the moduli set* $\{m_1, m_2, m_3\}$ *with* $m_1 = 2^{2n+1}-1$, $m_2 = 2^{2n}$, $m_3 = 2^n-1$, *the following hold true:*

$$|(m_1 m_2)^{-1}|_{m_3} = 1, \tag{8.1}$$

$$|(m_1 m_3)^{-1}|_{m_2} = 2^n + 1, \tag{8.2}$$

$$|(m_2 m_3)^{-1}|_{m_1} = 2^{2n+1} - 2^{n+2} - 5. \tag{8.3}$$

Proof: If it can be demonstrated that $|1 \times (m_1 m_2)|_{m_3} = 1$, then 1 is the multiplicative inverse of $(m_1 m_2)$ with respect to m_3:

$$\begin{aligned} R_1 &= \left|2^{2n}(2^{2n+1}-1)\right|_{2^n-1} = \left|(2^{2n+1}-1)\right|_{2^n-1} \\ &= \left|\left|2^{2n+1}\right|_{2^n-1} - |1|_{2^n-1}\right|_{2^n-1} = |2-1|_{2^n-1} = 1, \end{aligned}$$

thus Equation (8.1) holds true.

In the same way, if $|(2^n+1)\times(m_1m_3)|_{m_2}=1$, then (2^n+1) is the multiplicative inverse of (m_1m_3) with respect to m_2:

$$\begin{aligned} R_2 &= \left|(2^n+1)(2^{2n+1}-1)(2^n-1)\right|_{2^{2n}} \\ &= \left|(2^{2n}-1)(2^{2n+1}-1)\right|_{2^{2n}} \\ &= \left|2^{4n+1}-2^{2n}-2^{2n+1}+1\right|_{2^{2n}} \\ &= |0-0-0+1|_{2^{2n}}=1, \end{aligned}$$

thus Equation (8.2) holds true.

Again, if $\left|(2^{2n+1}-2^{n+2}-5)\times(m_2m_3)\right|_{m_1}=1$, then $(2^{2n+1}-2^{n+2}-5)$ is the multiplicative inverse of (m_2m_3) with respect to m_1:

$$\begin{aligned} R_3 &= \left|(2^{2n+1}-2^{n+2}-5)(2^{2n})(2^n-1)\right|_{2^{2n+1}-1} \\ &= \left|(2^{2n+1}-2^{n+2}-5)(2^{3n}-2^{2n})\right|_{2^{2n+1}-1} \\ &= \left|2^{3n}\left(2^{2n+1}\right)-2^{4n+2}-5\left(2^{3n}\right)\right. \\ &\quad \left.-2^{4n+1}+2^{3n+2}+5\left(2^{2n}\right)\right|_{2^{2n+1}-1} \\ &= \left|2^{3n}\left(2^{2n+1}\right)-3\left(2^{2n}\right)\left(2^{2n+1}\right)\right. \\ &\quad \left.-2^{3n}+5\left(2^{2n}\right)\right|_{2^{2n+1}-1} \\ &= \left|2^{3n}-3\left(2^{2n}\right)-2^{3n}+5\left(2^{2n}\right)\right|_{2^{2n+1}-1} \\ &= \left|2^{2n+1}\right|_{2^{2n+1}-1}=1, \end{aligned}$$

thus Equation (8.3) holds true.

The following important relations will be utilized in the subsequent theorem: Given the moduli set $\{m_1,m_2,m_3\}$ with $m_1=2^{2n+1}-1$, $m_2=2^{2n}$, $m_3=2^n-1$, the following hold true:

$$m_1 = 2m_2-1, \tag{8.4}$$

$$m_2 = m_3m_3+2m_3+1. \tag{8.5}$$

Theorem 8.3 *The decimal equivalent of the residues (x_1,x_2,x_3) with respect to the moduli set $\{m_1,m_2,m_3\}$ in the form $\{2^{2n+1}-1,2^{2n},2^n-1\}$, assuming $X\in[0,M-(m_2)^2+m_2)$, can be computed as follows:*

$$X=m_2\left\lfloor\frac{X}{m_2}\right\rfloor+x_2 \tag{8.6}$$

where

$$\left\lfloor \frac{X}{m_2} \right\rfloor = x_3 - x_2 + m_3 \left| (-2^{n+2} - 4)x_1 + 2^{n+1}x_2 + 2x_2 + 2^{n+1}x_3 + 2x_3 \right|_{m_1} \quad (8.7)$$

Proof: Since Equation (8.6) follows the basic integer division definition in RNS, which is always true, we only need to show the correctness of Equation (8.7). The CRT [111] for $k = 3$ is given by:

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M \quad (8.8)$$

Substituting Equations (8.1), (8.2), and (8.3) into Equation (8.8) we obtain the following:

$$X = \left| m_2 m_3 (2^{2n+1} - 2^{n+2} - 5)x_1 + m_1 m_3 (2^n + 1)x_2 + m_1 m_2 x_3 \right|_M ,$$

and by substituting Equations (8.4) and (8.5) in the above equation, we obtain:

$$\begin{aligned} X &= \left| (m_2 m_3)(2^{2n+1} - 2^{n+2} - 3)x_1 + \right. \\ &\quad \left. m_3(2m_2 m_2 - 1)x_2 + m_2(2m_2 m_2 - 1)x_3 \right|_M , \\ X &= \left| (m_2 m_3)(2^{2n+1} - 2^{n+2} - 5)x_1 + \right. \\ &\quad (2^{n+1} m_2 m_3 x_2 - 2^n m_3 x_2 + 2m_2 m_3 - m_3 x_2) \\ &\quad \left. +(2m_3 m_3 m_2 x_3 - m_3 m_3 x_3 + 2m_3 x_3 + x_3) \right|_M . \end{aligned} \quad (8.9)$$

Equation (8.9) can be further simplified by using the following lemma, presented in [131]:

$$|am_1|_{m_1 m_2} = m_1 |a|_{m_2} \quad (8.10)$$

Applying Equations (8.10) and (8.5) and also since $\left|2^{2n+1}\right|_{2^{2n+1}-1} = 1$, Equation (8.9) becomes:

$$\begin{aligned} X &= |-2^n m_3 x_2 - m_3 x_2 + m_2 x_3 + \\ &\quad m_2 m_3 \left| (-2^{n+2} - 4)x_1 + 2^{n+1} x_2 \right. \\ &\quad \left. +2x_2 + 2m_3 x_3 + 2^2 x_3 \right|_{m_1} \Big|_M \end{aligned} \quad (8.11)$$

Further simplifications give:

$$\begin{aligned} X &= \big|-m_2x_2 + x_2 + m_2x_3 + m_2m_3\,\big|(-2^{n+2}-4)x_1 + 2^{n+1}x_2 \\ &\quad +2x_2 + 2m_3x_3 + 2^2x_3\big|_{m_1}\Big|_M \\ &= \big|m_2x_3 - m_2x_2 + x_2 + m_2m_3\,\big|(2^{2n+1} - 2^{n+2} - 3)x_1 \\ &\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_M \end{aligned} \tag{8.12}$$

Dividing both sides of the above equation by m_2 and taking the floor with further simplifications, we have:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= \big|x_3 - x_2 + m_3\,\big|-2^{n+2}x_1 - 2^2x_1 + 2^{n+1}x_2 \\ &\quad +2x_2 + 2^{n+1}x_3 + 2x_3\big|_{m_1}\Big|_{m_1m_3} \\ &= \Big|x_3 - x_2 + m_3\,\Big||2^{2n+1}x_1\big|_{m_1} - 2^{n+2}x_1 \\ &\quad -3x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \\ &= \big|x_3 - x_2 + m_3\,\big|x_1 - 2^{n+2}x_1 - 3x_1 \\ &\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \\ &= \big|x_3 - x_2 + m_3\,\big|-2^{n+2}x_1 - 2x_1 \\ &\quad +2m_2x_2 + 2m_2x_3 + 2x_3|_{m_1}\big|_{m_1m_3} \end{aligned} \tag{8.13}$$

It can be seen that Equation (8.13) is the general expression of Equation (8.7), which is valid for the entire dynamic range, $[0, M-1]$. The next stage of the proof is to demonstrate that the corrective addition required for the calculation of the mod-m_1m_3 can be avoided in most of the cases.

By definition of modulus, we have:

$$\begin{aligned} 0 &\le \big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \le m_1 - 1 \\ 0 &\le m_3\,\big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \\ &\le m_1m_3 - m_3 \end{aligned} \tag{8.14}$$

Observing that $0 \le x_3 < m_3$ and $0 \le x_2 < m_2\,(= m_3 + 1)$,
and using (8.14), we have:

$$\begin{aligned} -m_3 &\le -x_2 \le x_3 - x_2 \\ &+ \quad m_3\,\big|-2^{n+2}x_1 - 2x_1 + 2m_2x_2 + 2m_2x_3 + 2x_3\big|_{m_1} \\ &< m_1m_3 - m_3 + m_3\,(= m_1m_3) \end{aligned} \tag{8.15}$$

Thus, one corrective addition of $m_1 m_3$ is required in order to obtain the correct result when $x_3 - x_2 + m_3 \left| -2^{n+2} x_1 - 2x_1 + 2m_2 x_2 + 2m_2 x_3 + 2x_3 \right|_{m_1} < 0$.

Further, we show that if we slightly restrict the RNS dynamic range, no corrective addition is required. For the numbers that require corrective addition, the following condition holds true:

$$\begin{aligned} -x_2 + m_1 m_3 &\leq \left\lfloor \frac{X}{m_2} \right\rfloor < m_1 m_3 \\ M - m_2 x_2 &\leq m_2 \left\lfloor \frac{X}{m_2} \right\rfloor < M \\ M - (m_2 - 1) m_2 &\leq X < M \\ M - (m_2)^2 + m_2 &\leq X < M. \end{aligned} \tag{8.16}$$

Therefore, the numbers within the interval $[0, M - (m_2)^2 + m_2)$ require no corrective addition and thus, Equation (8.7) holds true.

We can further reduce the hardware complexity of the reverse converter by simplifying Equation (8.7) using the following two properties [86].

Property 1: Modulo $(2^s - 1)$ multiplication of a residue number by 2^t, where s and t are positive integers, is equivalent to t bit circular left shifting.

Property 2: Modulo $(2^s - 1)$ of a negative number is equivalent to the one's complement of the number, which is obtained by subtracting the number from $(2^s - 1)$.

Suppose that Equation (8.7) is written as:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^n - 1)A \\ &= x_3 - x_2 + 2^n A - A, \end{aligned} \tag{8.17}$$

where

$$A = \left| u_0 + u_1 + u_2 + u_3 + u_4 \right|_{2^{2n+1}-1}. \tag{8.18}$$

For simplicity sake, let us represent Equation (8.17) as:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_1 + B_2 + B_3, \tag{8.19}$$

where

$$\begin{aligned} B_1 &= -x_2, \\ B_2 &= 2^n A + x_3, \\ B_3 &= -A. \end{aligned} \tag{8.20}$$

Let the binary representations of the residues be:

$$\begin{aligned} x_1 &= (x_{1,2n}x_{1,2n-1}...x_{1,1}x_{1,0}), \\ x_2 &= (x_{2,2n-1}x_{2,2n-2}...x_{2,1}x_{2,0}), \\ x_3 &= (x_{3,n-1}x_{3,n-2}...x_{3,1}x_{3,0}). \end{aligned}$$

In Equation (8.18), u_0, u_1, u_2, u_3, and u_4 are represented as follows:

$$\begin{aligned} u_0 &= \left|-2^{n+2}x_1\right|_{2^{2n+1}-1} \\ &= \left|-2^{n+2}(x_{1,2n}x_{1,2n-1}...x_{1,0})\right|_{2^{2n+1}-1} \\ &= \left|-2^{n+2}(\underbrace{x_{1,2n}x_{1,2n-1}...x_{1,n-1}}_{n+2}\underbrace{x_{1,n-2}...x_{1,0}}_{n-1})\right|_{2^{2n+1}-1} \\ &= -(\underbrace{\cdots x_{1,0}x_{1,2n}x_{1,2n-1}\cdots}_{2n+1}) \\ &= (\underbrace{\overline{x}_{1,n-2}\overline{x}_{1,n-3}...\overline{x}_{1,1},\overline{x}_{1,0}}_{n-1}\underbrace{\overline{x}_{1,2n}\overline{x}_{1,2n-1}...\overline{x}_{1,n-1}}_{n+2}), \end{aligned}$$

$$\begin{aligned} u_1 &= \left|-2^2x_1\right|_{2^{2n+1}-1} \\ &= \left|-2^2(x_{1,2n}x_{1,2n-1}...x_{1,0})\right|_{2^{2n+1}-1} \\ &= -(\underbrace{x_{1,2n-1}\cdots x_{1,0}x_{1,2n}}_{2n+1}) \\ &= (\underbrace{\overline{x}_{1,2n-2}\cdots\overline{x}_{1,0}\overline{x}_{1,2n}\overline{x}_{1,2n-1}}_{2n+1}), \end{aligned}$$

$$\begin{aligned} u_2 &= \left|2^{n+1}x_2\right|_{2^{2n+1}-1} \\ &= \left|2^{n+1}(x_{2,2n}x_{2,2n-1}...x_{2,0})\right|_{2^{2n+1}-1} \\ &= \left|2^n(x_{2,2n}x_{2,2n-1}...x_{2,0}0)\right|_{2^{2n+1}-1} \\ &= \left|2^n\underbrace{x_{2,2n-1}x_{2,2n-2}\cdots x_{2,n}}_{n}\underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}0}_{n+1}\right|_{2^{2n+1}-1} \\ &= \underbrace{x_{2,n-1}x_{2,n-2}\cdots x_{2,0}0}_{n+1}\underbrace{x_{2,2n-1}x_{2,2n-2}\cdots x_{2,n}}_{n}, \end{aligned}$$

$$\begin{aligned} u_3 &= |2x_2|_{2^{2n+1}-1} \\ &= |(x_{2,2n-1}x_{2,2n-2}...x_{2,0})|_{2^{2n+1}-1} \\ &= (\underbrace{x_{2,2n-1}x_{2,2n-2}\cdots x_{2,0}0}_{2n+1}), \end{aligned}$$

$$\begin{aligned} u_4 &= \left|2^{n+1}x_3 + 2x_3\right|_{2^{2n+1}-1} \\ &= \left|\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}\underbrace{00\cdots 0}_{n+1} + \underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}0}_{n+1}\right|_{2^{2n+1}-1} \\ &= (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}x_{3,n-1}x_{3,n-2}\cdots x_{3,0}0}_{2n+1}. \end{aligned}$$

Given the binary representation:

$$A = (\underbrace{a_{2n}a_{2n-1}\cdots a_1a_0}_{2n+1}),$$

B_2 can be written as:

$$\begin{aligned} B_2 &= (\underbrace{a_{2n}a_{2n-1}\cdots a_1a_0}_{2n+1}\underbrace{00\cdots 0}_{n}) + (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\ &= (\underbrace{a_{2n}a_{2n-1}\cdots a_0x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{3n+1}) \end{aligned} \tag{8.21}$$

In Equation (8.19), in order to carry out the summation, B_1 and B_3 must have equal number of bits, i.e., $(3n+1)$-bits, as B_2. They can be represented as:

$$\begin{aligned} B_1 &= -x_2 \\ &= -(\underbrace{000\cdots 0}_{n+1}\underbrace{x_{2,2n-1}x_{2,2n-2}\cdots x_{2,0}}_{2n}) \\ &= (\underbrace{111\cdots 11}_{n+1}\underbrace{\overline{x}_{2,2n-1}\overline{x}_{2,2n-2}\cdots \overline{x}_{2,0}}_{2n}), \end{aligned} \tag{8.22}$$

$$\begin{aligned} B_3 &= -A \\ &= -(\underbrace{000\cdots 0}_{n}\underbrace{a_na_{n-1}\cdots a_0}_{2n+1}) \\ &= (\underbrace{111\cdots 11}_{n}\underbrace{\overline{a}_n\overline{a}_{n-1}\cdots \overline{a}_0}_{2n+1}). \end{aligned} \tag{8.23}$$

8.2 Domain Restriction Problem

We resolve the dynamic range limitation problem for the moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$ in this section. In Equation (8.7), if $x_3 - x_2 + m_3 \left|(-2^{n+2}-4)x_1 + 2^{n+1}x_2 + 2x_2 + 2^{n+1}x_3 + 2x_3\right|_{m_1} < 0$, then $\left|-2^{n+2}x_1 - 2^2x_1 + 2^{n+1}x_2 + 2x_2 + 2^{n+1}x_3 + 2x_3\right|_{m_1} < 2^n + 1$ since $x_2 \leq 2^{2n} - 1 = (2^n - 1)(2^n + 1)$ and $m_3 = 2^n - 1$. Thus, we have this negative value iff:

$$x_2 > x_3 \wedge \left|-2^{n+2}x_1 - 2^2x_1 + 2^{n+1}x_2 + 2x_2 + 2^{n+1}x_3 + 2x_3\right|_{m_1} < 2^n + 1 \tag{8.24}$$

For this case of condition (8.24), only one corrective addition of m_1m_3 is required as shown in the previous section. Therefore, Equation (8.7) can be written as:

$$\left\lfloor \frac{X}{m_2} \right\rfloor = x_3 - x_2 + m_1m_3 \tag{8.25}$$

Equation (8.25) can be simplified as follows:

$$\begin{aligned} \left\lfloor \frac{X}{m_2} \right\rfloor &= x_3 - x_2 + (2^{2n+1} - 1)(2^n - 1) \\ &= x_3 - x_2 + 2^{3n+1} - 2^{2n+1} - 2^n + 1 \\ &= x_3 - x_2 + 2^{3n+1} - 2^{2n+1} + 2^n \end{aligned} \tag{8.26}$$

Equation (8.26) may be simplified as follows:

$$\begin{aligned}
B_5 &= -x_2 \\
&= \underbrace{(\underbrace{111\cdots11}_{n+2}\underbrace{\overline{x}_{2,2n-1}\overline{x}_{2,2n-2}\cdots\overline{x}_{2,0}}_{2n})}_{3n+2}, && (8.27)\\
B_4 &= 2^n + x_3 \\
&= (\underbrace{100\cdots00}_{n+1}) + (\underbrace{x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n}) \\
&= (\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}), && (8.28)\\
B_6 &= B_4 + 2^{3n+1} \\
&= (\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}) + (\underbrace{100\cdots0}_{3n+2}) \\
&= (\underbrace{100\cdots0}_{2n+1})(\underbrace{1x_{3,n-1}x_{3,n-2}\cdots x_{3,0}}_{n+1}), && (8.29)\\
B_7 &= -2^{2n+1} \\
&= -(\underbrace{100\cdots00}_{2n+2}) \\
&= -(\underbrace{00\cdots0}_{n})(\underbrace{100\cdots00}_{2n+2}) \\
&= (\underbrace{11\cdots1}_{n})(\underbrace{011\cdots1}_{2n+2}). && (8.30)
\end{aligned}$$

Therefore,

$$\left\lfloor \frac{X}{m_2} \right\rfloor = B_5 + B_6 + B_7 \qquad (8.31)$$

8.3 MRC Based Converter

In this section, we present an MRC technique, which is valid for the entire dynamic range for the proposed moduli set. Here, the intention is to verify if an MRC technique will be a better conversion approach when compared to the entire dynamic CRT technique proposed for the $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$ moduli set in the previous sections.

Theorem 8.4 *Given the moduli set* $\{m_1, m_2, m_3\}$ *with* $m_1 = 2^n - 1, m_2 =$

$2^{2n}, m_3 = 2^{2n+1} - 1$, *the following holds true:*

$$|(m_1)^{-1}|_{m_2} = 2^{2n} - 2^n - 1, \tag{8.32}$$
$$|(m_2)^{-1}|_{m_3} = 2, \tag{8.33}$$
$$|(m_1)^{-1}|_{m_3} = 2^{2n+1} - 2^{n+1} - 3. \tag{8.34}$$

Proof:
If it can be shown that $|2^{2n} - 2^n - 1 \times m_1|_{m_2} = 1$, then $2^{2n} - 2^n - 1$ is the multiplicative inverse of m_1 with respect to m_2:

$$\begin{aligned} R_4 &= \left|(2^{2n} - 2^n - 1)(2^n - 1)\right|_{2^{2n}} \\ &= |2^{3n} - 2^{2n} - 2^{2n} + 1|_{2^{2n}} \\ &= |2^{3n} - 2(2^{2n}) + 1|_{2^{2n}} \\ &= |0 - 0 + 1|_{2^{2n}} = 1, \end{aligned}$$

thus Equation (8.32) holds true.

In the same way, if $|2 \times m_2|_{m_3} = 1$, then 2 is the multiplicative inverse of m_2 with respect to m_3:

$$\begin{aligned} R_5 &= \left|2(2^{2n})\right|_{2^{2n+1}-1} = 1, \\ &= \left|2^{2n+1}\right|_{2^{2n+1}-1} = 1, \end{aligned}$$

thus Equation (8.33) holds true.

Similarly,

$$\begin{aligned} R_6 &= \left|(2^{2n+1} - 2^{n+1} - 3)(2^n - 1)\right|_{2^{2n+1}-1} \\ &= \left|2^n(2^{2n+1}) - 2(2^{2n+1}) - 2^n + 3\right|_{2^{2n+1}-1} \\ &= |2^n - 2 - 2^n + 3|_{2^{2n+1}-1} = 1, \end{aligned}$$

thus Equation (8.34) also holds true.

For 3-moduli set the MRC presented in [141] is given by:

$$X = a_1 + a_2 m_1 + a_3 m_1 m_2 \tag{8.35}$$

where the Mixed Radix Digits (MRDs) are given as follows:

$$\begin{aligned} a_1 &= x_1 \\ a_2 &= \left|(x_2 - a_1)\left|m_1^{-1}\right|_{m_2}\right|_{m_2} \\ a_3 &= \left|\left((x_3 - a_1)\left|m_1^{-1}\right|_{m_3} - a_2\right)\left|m_2^{-1}\right|_{m_3}\right|_{m_3} \end{aligned} \tag{8.36}$$

Using Equations (8.32) and (8.33) in Equation (8.36), the MRDs a_1, a_2, and a_3 can be represented as

$$a_1 = x_1, \tag{8.37}$$

$$\begin{aligned} a_2 &= \left|(2^{2n} - 2^n - 1)(x_2 - x_1)\right|_{2^{2n}} \\ &= |2^{2n}x_2 - 2^n x_2 - x_2 - 2^{2n}x_1 + 2^n x_1 + x_1|_{2^{2n}} \\ &= |-2^n x_2 - x_2 + 2^n x_1 + x_1|_{2^{2n}}, \end{aligned} \tag{8.38}$$

and

$$\begin{aligned} a_3 &= \left|((x_3 - x_1)(2^{2n+1} - 2^{n+1} - 3) - a_2)(2)\right|_{2^{2n+1}-1} \\ &= \left|-2^{n+2}x_3 - 4x_3 + 2^{n+2}x_1 + 4x_1 - 2a_2\right|_{2^{2n+1}-1}. \end{aligned} \tag{8.39}$$

The hardware realization of Equation (8.35) can be further simplified using the two properties presented in Section 8.1. Let the binary representations of the residues be:

$$\begin{aligned} x_1 &= (x_{1,n-1}x_{1,n-2}...x_{1,1}x_{1,0}), \\ x_2 &= (x_{2,2n-1}x_{2,2n-2}...x_{2,1}x_{2,0}), \\ x_3 &= (x_{3,2n}x_{3,2n-1}...x_{3,1}x_{3,0}). \end{aligned}$$

Given that Equation (8.38) is represented as

$$a_2 = |v_{1,1} + v_{1,2} + v_{1,3}|_{2^{2n}} \tag{8.40}$$

where

$$\begin{aligned} v_{1,1} &= |2^n x_1 + x_1|_{2^{2n}} \\ &= \left|\underbrace{x_{1,n-1}x_{1,n-2}\cdots x_{1,0}}_{n}\underbrace{00\cdots0}_{n} + \underbrace{x_{1,n-1}x_{1,n-2}\cdots x_{1,0}}_{n}\right|_{2^{2n}} \\ &= (\underbrace{x_{1,n-1}x_{1,n-2}\cdots x_{1,0}x_{1,n-1}x_{1,n-2}\cdots x_{1,0}}_{2n}), \end{aligned}$$

$$\begin{aligned}
v_{1,2} &= |-2^n x_2|_{2^{2n}} \\
&= |-2^n (x_{2,2n-1} x_{2,2n-2} ... x_{2,0})|_{2^{2n}} \\
&= \left| -(\underbrace{x_{2,2n-1} x_{2,2n-2} ... x_{2,0}}_{2n} \underbrace{00...0}_{n}) \right|_{2^{2n}} \\
&= \left| \underbrace{\overline{x}_{2,2n-1} \overline{x}_{2,2n-2} \cdots \overline{x}_{2,n}}_{n} \underbrace{\overline{x}_{2,n-1} \overline{x}_{2,n-2} \cdots \overline{x}_{2,0}}_{n} \underbrace{\overline{0}\overline{0} \cdots \overline{0}}_{n} \right|_{2^{2n}} \\
&= \underbrace{\overline{x}_{2,n-1} \cdots \overline{x}_{2,0} 11 \cdots 1}_{2n},
\end{aligned}$$

and

$$\begin{aligned}
v_{1,3} &= |-x_2|_{2^{2n}} \\
&= \left| -(\underbrace{x_{2,2n-1} x_{2,2n-2} ... x_{2,0}}_{2n}) \right|_{2^{2n}} \\
&= \underbrace{\overline{x}_{2,2n-1} \overline{x}_{2,2n-2} \cdots \overline{x}_{2,0}}_{2n}
\end{aligned}$$

Representing Equation (8.39) as:

$$a_3 = |v_{2,1} + v_{2,2} + v_{2,3} + v_{2,4}|_{2^{2n+1}-1} \tag{8.41}$$

where

$$\begin{aligned}
v_{2,1} &= \left| -2^{n+2} x_3 \right|_{2^{2n+1}-1} \\
&= \left| -2^{n+2} (x_{3,2n} x_{3,2n-1} ... x_{3,0}) \right|_{2^{2n+1}-1} \\
&= \left| 2^{n+2} (\underbrace{\overline{x}_{3,2n} \overline{x}_{3,2n-1} ... \overline{x}_{3,n+2}}_{n-1} \underbrace{\overline{x}_{3,n+1} \overline{x}_{3,n} ... \overline{x}_{3,0}}_{n+2}) \right|_{2^{2n+1}-1} \\
&= \left| \underbrace{\overline{x}_{2,2n-1} \overline{x}_{2,2n-2} \cdots \overline{x}_{2,n}}_{n} \underbrace{\overline{x}_{2,n-1} \overline{x}_{2,n-2} \cdots \overline{x}_{2,0}}_{n} \underbrace{\overline{0}\overline{0} \cdots \overline{0}}_{n} \right|_{2^{2n}} \\
&= \underbrace{\overline{x}_{3,n+1} \overline{x}_{3,n} ... \overline{x}_{3,0}}_{n+2} \underbrace{\overline{x}_{3,2n} \overline{x}_{3,2n-1} ... \overline{x}_{3,n+2}}_{n-1},
\end{aligned}$$

$$\begin{aligned} v_{2,2} &= |-4x_3|_{2^{2n+1}-1} \\ &= \left| 2^2 \underbrace{\overline{x}_{3,2n}\overline{x}_{3,2n-1}...\overline{x}_{3,0}}_{2n+1} \right|_{2^{2n+1}-1} \\ &= \underbrace{\overline{x}_{3,1}\overline{x}_{3,0}\overline{x}_{3,2n}\overline{x}_{3,2n-1}...\overline{x}_{3,2}}_{2n+1}, \end{aligned}$$

$$\begin{aligned} v_{2,3} &= \left|2^{n+2}x_1 + 4x_1\right|_{2^{2n+1}-1} \\ &= \left| \underbrace{(x_{1,n-1}x_{1,n-2}\cdots x_{1,0})}_{n} \underbrace{00\cdots0}_{n+2} + \underbrace{x_{1,n-1}x_{1,n-2}\cdots x_{1,0}00}_{n+2} \right|_{2^{2n+1}-1} \\ &= (0\underbrace{x_{1,n-1}x_{1,n-2}\cdots x_{1,0}x_{1,n-1}x_{1,n-2}\cdots x_{1,0}}_{2n+1}, \end{aligned}$$

$$\begin{aligned} v_{2,4} &= |-2a_2|_{2^{2n+1}-1} \\ &= \left| -(\underbrace{a_{2,2n-1}a_{2,2n-2}...a_{2,0}0}_{2n+1}) \right|_{2^{2n+1}-1} \\ &= \underbrace{\overline{a}_{2,2n-1}\overline{a}_{2,2n-2}\cdots\overline{a}_{2,0}1}_{2n+1} \end{aligned}$$

Equation (8.35) can be simplified as follows

$$\begin{aligned} X &= a_1 + a_2m_1 + a_3m_1m_2 \\ &= a_1 + 2^na_2 + 2^{3n}a_3 - 2^{2n}a_3 - a_2. \end{aligned} \tag{8.42}$$

The hardware realization of Equation (8.42) can be simplified as follows

$$X = a_4 + a_5 + a_6 \tag{8.43}$$

where

$$\begin{aligned} a_4 &= a_1 + 2^na_2 + 2^{3n}a_3 \\ &= (\underbrace{a_{1,n-1}a_{1,n-2}...a_{1,0}}_{n}) + (\underbrace{a_{2,2n-1}a_{2,2n-2}...a_{2,0}}_{2n}\underbrace{00...0}_{n}) \\ &\quad +(\underbrace{a_{3,2n}a_{3,2n-1}...a_{3,0}}_{2n+1}\underbrace{00...0}_{3n}) \\ &= \underbrace{a_{3,2n}...a_{3,0}a_{2,2n-1}...a_{2,0}a_{1,n-1}...a_{1,0}}_{5n+1}, \end{aligned} \tag{8.44}$$

$$\begin{aligned} a_5 &= -2^{2n}a_3 \\ &= -(\underbrace{00...0}_{n}\underbrace{a_{3,2n}...a_{3,0}}_{2n+1}\underbrace{00...0}_{2n}) \\ &= \underbrace{11...1}_{n}\underbrace{\overline{a}_{3,2n}...\overline{a}_{3,0}}_{2n+1}\underbrace{11...1}_{2n} \end{aligned} \tag{8.45}$$

$$\begin{aligned} a_6 &= -a_2 \\ &= -(\underbrace{a_{2,2n-1}a_{2,2n-2}...a_{3,0}}_{2n}) \\ &= -(\underbrace{00...0}_{3n+1}\underbrace{a_{2,2n-1}a_{2,2n-2}...a_{2,0}}_{2n}) \\ &= \underbrace{11...1}_{3n+1}\underbrace{\overline{a}_{2,2n-1}...\overline{a}_{2,0}}_{2n} \end{aligned} \tag{8.46}$$

8.4 Descriptions of the Proposed Hardware Structures

In this section, the structures that result from the RNS to Binary conversion methods previously described are presented.

8.4.1 CRT Conversion Techniques

We start by describing the structure that results from the limited range CRT approach, which is based on Equations (8.18) and (8.19). As it can be seen from Figure 8.1, u_0, u_1, u_2, u_3, and u_4 are added by Carry Save Adders (CSAs) with end-around carries (EACs) producing the values s_3 and c_3. These values must be added modulo $2^{2n+1}-1$ in order to obtain A, i.e., with a one's complement adder, namely a CPA with EAC. B_2 is easily obtained by concatenating the operand x_3 with the n-bit left shift of A. This concatenation does not require any additional hardware resources. The three operands B_1, B_2, and B_3 are added using a CSA with EAC. It should be noted that in order to make B_1 and B_3 $(3n+1)$-bit numbers, 1's are appended to the result of complementations, as given in Equations (8.22) and (8.23). Thus, the addition of the $(n+1)$ Most Significant Bits (MSB) performed in this CSA can be performed by Half Adders (HAs). Moreover, since the n MSB of the CSA all have two inputs equal to 1, the final one's complement adder will always generate an EAC. Taking this into consideration, the one's complement adder can be reduced to a normal CPA with a constant carry-in equals to 1. The final result, computed

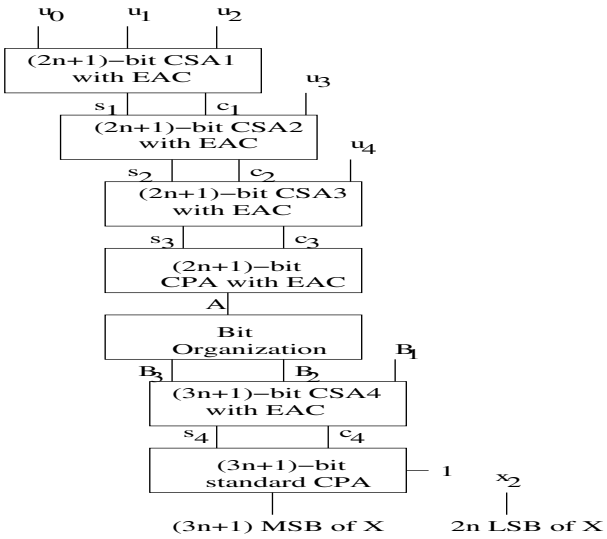


Figure 8.1: Proposed Reverse Converter I

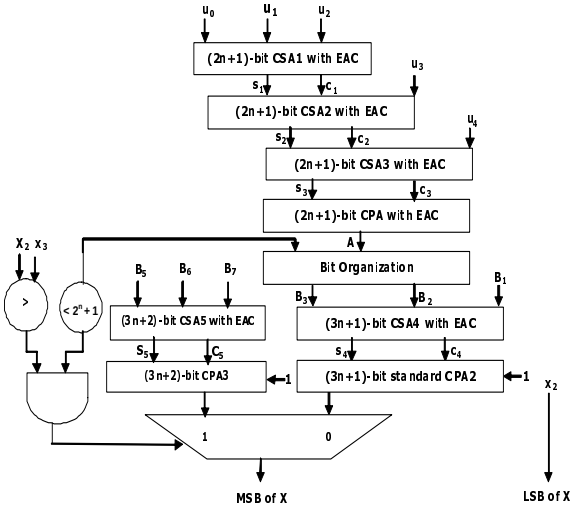


Figure 8.2: Proposed Reverse Converter-III

from Equation (8.6) is obtained simply by a shift and a concatenation operation, which do not require any additional hardware resources.

Given that the numbers that fall outside the range $[0, M-(m_2^2-2m_2+1))$ may be of interest, we resolve the domain restriction problem and propose a second converter, which is based on Equations (8.22), (8.23), and (8.31). The hardware implementations of the second proposal, which is valid for the entire dynamic range $[0, M-1]$, is depicted in Figure 8.2.

8.4.2 Entire Dynamic Range MRC Technique

The hardware structure that results from the MRC approach is depicted in Figure 8.3 and it is based on Equations (8.40), (8.41), and (8.43). In Equation (8.40), the operands are added using a $2n$-bit CSA with EAC, generating s_1 and c_1, which are added by a $2n$-bit CPA with EAC. Similarly, 2-levels of CSAs, followed by a $(2n+1)$-bit one's complement adder, are used to add the 4-operands in Equation (8.41). In Figure 8.3, Bit org1 depicts the bit reorganization used to obtain $v_{2,4}$. This is simply the result of complementations of 1-bit right shift of a_2. However, a_4 is simply obtained by concatenating the three operands a_3, a_2, and a_1, which are $(2n+1)$-bit, $2n$-bit, and n-bit, respectively. This concatenation is labeled as Bit org3 and does not require any additional hardware resources. In order to perform the addition described in

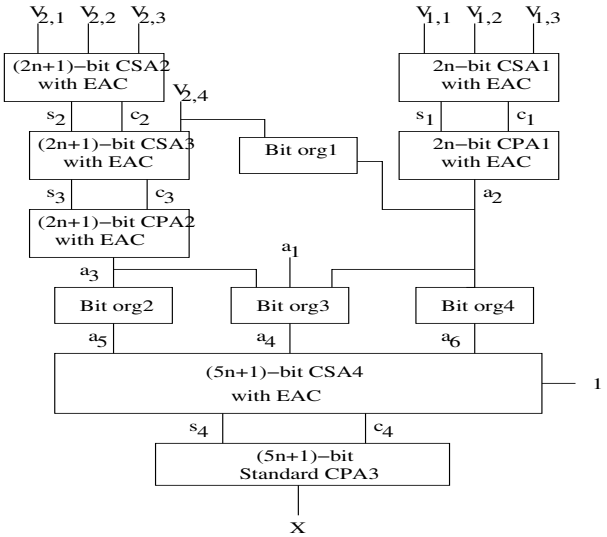


Figure 8.3: Proposed Reverse Converter II

Equation (8.43), which is identical to what we performed on the CRT conver-

Table 8.1: Area and Delay Comparisons

	FA	HA	Delay
MOL	$2.25D - 0.25$	$0.5D + 1.5$	$(2.5D + 2.5)t_{FA}$
KAG	$2.25D - 0.25$	$1.25D + 2.75$	$(1.75D + 5.25)t_{FA}$
Proposed CI	$2.4D - 0.4$	$0.4D + 3.6$	$(1.4D + 5.6)t_{FA}$
Proposed CII	$3D - 1$	$0.8D + 2.2$	$(2.6D + 4.4)t_{FA}$
Proposed CIII	$3.4D + 3.6$	$0.6D + 2.4$	$(1.4D + 7.6)t_{FA}$

sion structure, the two operands a_5 and a_6 must be expanded to $(5n + 1)$-bit numbers since a_4 is a $(5n+1)$-bit number. The final adder can be simplified to a standard CPA (CPA3) with a constant carry-in equals to 1, in a similar way to what has been done for the CRT.

8.5 Performance Evaluation

In order to evaluate the performance of the proposed reverse converters, we compare them with similar best known state of the art reverse converters in [88] and [39]. In order to properly perform this evaluation, for the same dynamic range, the reverse conversion structures are theoretically analyzed, in terms of gates, and experimentally by implementating them on an FPGA. We note that in Table 8.1, D stands for the dynamic range. In the table, we have the converters CI, CII, and CIII as the converters proposed in this chapter, MOL and KAG as the converters proposed in [88] and [39], respectively. From the results of the theoretical analysis, presented in Table 8.1, it is clear that the proposed CI is clearly more advantageous than the proposed CII and CIII, both in terms of area and delay. However, CI is not valid for the entire dynamic range whereas CII and CIII cover the entire dynamic range. Thus, it is more appropriate to compare CI with KAG, which does not also cover the entire dynamic range. The theoretical analysis results suggest that CI is faster than KAG with small area penalty. CIII outperforms CII in all terms. Thus, we compare CIII with MOL, which uses the $\{2^{2n+1} - 1, 2^n, 2^n - 1\}$ moduli set and can be seen clearly from Table 8.1 that CIII performs reverse conversion faster than MOL with additional area cost. This is expected since the moduli set herein proposed is identical to the moduli set in [88] with a binary channel twice the size. This implies that for the same dynamic range the size of n can be smaller, thus units with smaller length are used.

The FPGA implementation results presented in Tables 8.2, 8.3, and 8.4, val-

Table 8.2: Area Cost Comparison

Dynamic Range	Area-CI	Area-CII	Area-CIII	Area-MOL	Area-KAG
16	61	80	75	67	71
24	103	142	136	107	107
32	124	167	164	135	141
48	206	286	273	202	210
64	264	369	351	280	276

Table 8.3: Conversion Speed Comparison

Dynamic Range	Delay-CI	Delay-CII	Delay-CIII	Delay-MOL	Delay-KAG
16	31.342	37.962	32.306	34.418	34.048
24	43.874	48.127	45.522	45.600	37.129
32	39.689	50.871	40.966	48.908	40.319
48	47.595	59.125	49.586	56.400	44.010
64	49.477	64.811	49.465	59.759	49.180

idate the theoretical analysis results. In comparison with KAG, these results suggest that CI is able to improve the conversion computation time by about 6% while reducing the required hardware area by more than 3%. Regarding the proposed CII reverse converter, CIII improves the conversion speed by about 17% and reduces the hardware resources by more than 4%. For this reason, in the following, we only compare CIII with MOL.

When analyzing the FPGA implementation results, depicted in Figures 8.4 and 8.5 and Tables 8.2 and 8.3, the performance and area advantages of the

Table 8.4: Area-Time^2 Comparison

Dynamic Range	AT^2-CI	AT^2-CII	AT^2-CIII	AT^2-MOL	AT^2-KAG
16	59922	115289	78276	79368	82308
24	198268	328902	281826	222492	147506
32	195327	432172	275227	322919	229213
48	466649	999789	671245	642554	406745
64	646265	1549972	858822	999919	667554

proposed CRT based reverse converter (CIII) are confirmed to be significant when compared with the proposed MRC based converter (CII) and also with similar state of the art MRC based reverse converter (MOL).

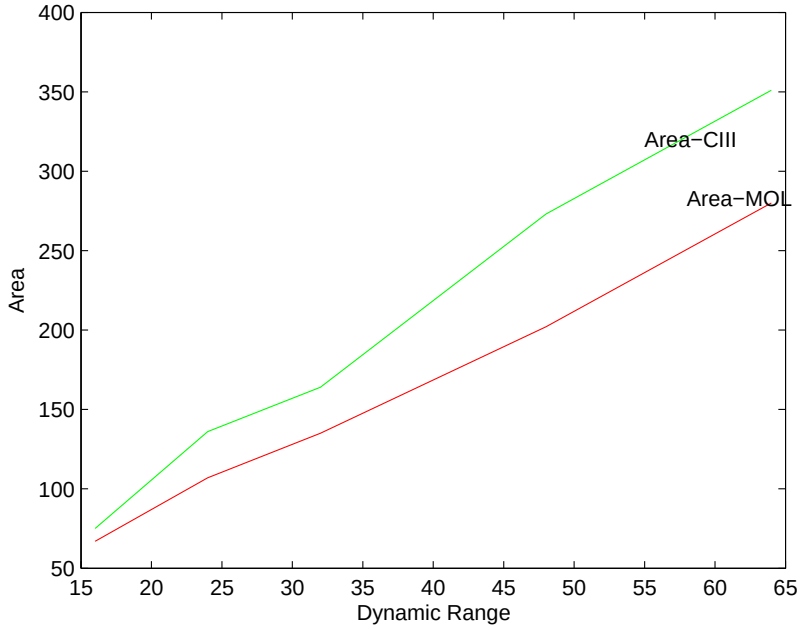


Figure 8.4: The Converters Area Comparisons

Experimental results indicate that the proposed CIII is about 11% faster and requires about 22% more hardware resources than the MOL. Additionally, the AT^2 metric indicates that the proposed CIII is 5% better than the MOL. Due to the fact that the moduli set utilized by the converter MOL underutilizes the binary channel, the moduli set proposed here has alot of advantages also with respect to other RNS arithmetic operations.

From these results it can be concluded that the proposed moduli set and respective CRT based reverse converter (CIII) are able to improve the performance of RNS computation.

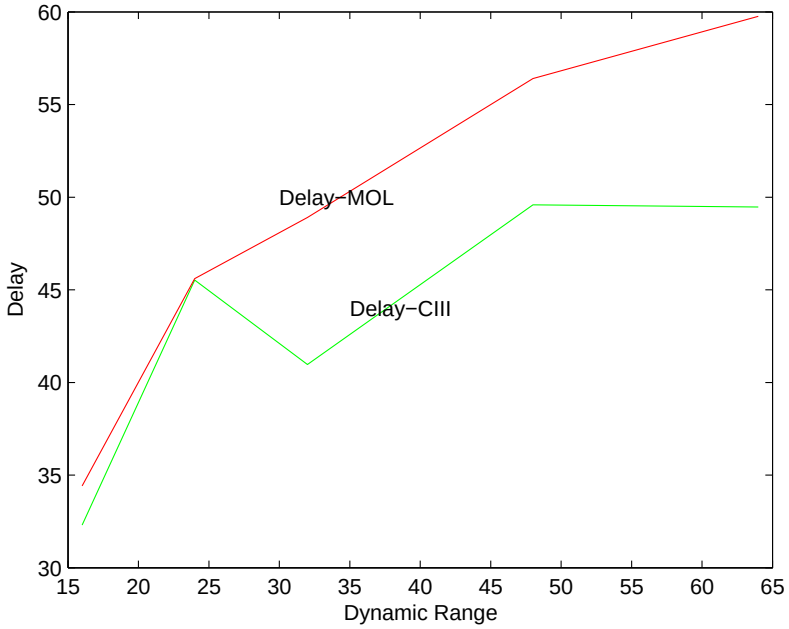


Figure 8.5: Conversion Delay Comparisons

8.6 Conclusion

In this chapter, three reverse converters for the new moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$ are proposed. Two of the proposed reverse converters are based on the traditional CRT while the other is based on the MRC. The CRT based reverse converters are better in terms of both area and delay, when compared to the MRC based reverse converter. Therefore, it can be inferred that the CRT is a better approach to design reverse converters for this class of moduli sets.The proposed reverse converters are memoryless and adder based and can be easily realized in VLSI circuits. We performed both theoretical and experimental evaluation of our proposal. The theoretical analysis shows the advantages of our scheme, which is supported by the experimental results. The synthesis results are obtained and suggest that, CI is able to improve the conversion computation time by about 6% while reducing the required hardware area by more than 3%. CIII improves the conversion speed by about 17% and reduces the hardware resources by more than 4%, regarding the proposed

CII reverse converter. The proposed CIII is about 11% faster and requires about 22% more hardware resources than the MOL. The moduli set proposed here has significant advantages also with respect to other RNS arithmetic operations, since the moduli set utilized by the converter MOL underutilizes the binary channel while the proposed moduli set does not.

This chapter concludes the presentation of our contributions. The next chapter contains the overall conclusions of our investigation. We will summarize our main contributions and we will propose some future research directions.

Chapter 9

Conclusions

In this thesis, we focussed on building effective Residue Number System (RNS) to decimal/binary converters based on either Mixed Radix Conversion (MRC) or on the Chinese Remainder Theorem (CRT). This is an important issue, since, while RNS computer arithmetic is theoretically faster than the standard one, this conversion imposes an overhead and limits the RNS utilization in the design of processors. Since this type of conversion is also required during the calculations in order to determine operand sign, overflow, etc., the utilization of RNS has been mostly restricted to special purpose digital signal processors. We proposed efficient reverse conversion techniques that result in fast hardware implementations potentially applicable for the design of general purpose processors.

The remainder of this chapter is organized as follows. Section 9.1 summarizes the work and results presented in this thesis. Section 9.2 highlights the main contributions of this thesis, while Section 9.3 presents some future directions and open issues, which are worthy of further investigation in the RNS arithmetic domain.

9.1 Summary

In this thesis, we proposed a number of efficient reverse converters based on either the CRT or MRC. The work presented in this thesis can be summarized as follows:

- In Chapter 2, we introduced RNS to MRC technique, which addresses

the computation of Mixed Radix Digits in such a way that enables the MRC parallelization. Given an RNS with a set of relatively prime integer moduli $\{m_i\}_{i=1,k}$, the key idea behind this proposed MRC technique is to maximize the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. For an n-digit RNS number $X = (x_1, x_2, x_3, ..., x_k)$ this method requires k iterations. However, at iteration i, the modulo-m_i units are utilized for the calculation of the MR digit a_i, while the other modulo units are calculating intermediate results required in further iterations. Given that one such iteration can be seen as one single operation in the RNS processor functional units, our approach results in an RNS to MR conversion with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, while the traditional MRC and many other state of the art MRCs based techniques exhibit an asymptotic complexity in the order of $O\left(k^2\right)$. More in particular, the utilization of our technique achieved 33.33% and 5.26% reductions with moduli sets of length four when compared with MRC and the improved MRC in [141], respectively. For moduli sets of length ten, our proposal achieved 66.67% and 38.64% reductions when compared with MRC and the improved MRC in [141], respectively.

- In Chapter 3, we generalized a previously proposed technique that was restricted to 5-moduli set such that it can be utilized in conjunction with any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$. Next, we simplified the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. For an k-digit RNS number $X = (x_1, x_2, x_3, ..., x_k)$ the proposed method takes at most k iterations. Each iteration, except the first one, requires two multiplications and one parallel subtraction over all the mod-m_i ways of the RNS adder. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, while the traditional MRC technique exhibits an asymptotic complexity in the order of $O\left(k^2\right)$. In particular, the utilization of our technique, for 3-moduli and 10-moduli RNS results in the reduction of the total number of arithmetic operations required by the conversion process with 40.00% and 77.16%, respectively, when compared to the MRC in [139].

- In Chapter 4, we proposed reverse converters for moduli sets with common factors. First, we assumed a general moduli set $\{m_i\}_{i=1,3}$, where $m_1 > m_2 > m_3$, with the dynamic range $M = \prod_{i=1}^{3} m_i$, and intro-

duced a modified CRT that requires mod-m_3 instead of mod-M calculations. Subsequently, we further simplified the conversion process by focussing on $\{2n+2, 2n+1, 2n\}$ moduli set, which has a common factor of 2. We introduced in a formal way a CRT based approach for this case, which requires the conversion of the $\{2n+2, 2n+1, 2n\}$ moduli set into a moduli set with relatively prime moduli, i.e., $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, when n is even, $n \geq 2$ and $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, when n is odd, $n \geq 3$. We demonstrated that such a conversion can be easily done and doesn't require the computation of any multiplicative inverses. We further simplified the 3-moduli CRT for the specific case of $\{2n+2, 2n+1, 2n\}$ moduli set. For this case the proposed CRT requires 4 additions, 4 multiplications, and all the operations are mod-m_3 in case n is even and mod-$\frac{m_3}{2}$ if n is odd. This outperforms state of the art converters in terms of required operations. Second, we proposed another new reverse converter for the moduli set $\{2n+2, 2n+1, 2n\}$ for any integer $n > 0$. We simplified the traditional CRT in order to obtain a reverse converter that uses mod-n instead of mod-$(2n+2)(2n)$ or mod-$2n$ required by other state of the art equivalent converters. Next, we presented a low complexity implementation that does not require the explicit calculation of modulo operation in the conversion process as it is normally the case in the traditional CRT and other state of the art equivalent converters. In terms of area, our proposal requires four 2:1 adders and 2 multipliers while the best state of the art equivalent converter [44] requires one 3:1 adder, two 2:1 adders, and four multipliers. In terms of critical path delay, our scheme requires 3 additions and 1 multiplication with mod-n operations whereas the converter in [44] requires 2 additions and 2 multiplications with mod-$2n$ operations. Third, we proposed a general 4-moduli RNS conversion scheme and then presented a compact form of multiplicative inverses valid for both n odd or even for the moduli set $\{2n+3, 2n+2, 2n+1, 2n\}$. This increases the dynamic range and the processing parallelism enabling efficient reverse conversion. The proposed CRT utilizes the same or slightly larger area when compared to other existing techniques but all the operations are mod-m_4. This outperforms state of the art CRTs in terms of the magnitude of the numbers involved in the calculation and due to this fact, our proposal results in less complex adders and multipliers.

- In Chapter 5, we proposed a new reverse converter for the moduli set $\{2n+1, 2n, 2n-1\}$. First, we simplified the traditional CRT in order to obtain a reverse converter that utilizes mod-$(2n-1)$ operations instead

of mod-$(2n)(2n-1)$ and $(2n+1)(2n-1)$ operations required by the converters in [41] and [2], respectively. Next, we transformed the needed corrective addition of M into a corrective m_3 addition. We then presented a novel low complexity implementation that does not require the explicit use of modulo operation in the conversion process. The performance of the proposed converter is evaluated both theoretically, in terms of the required number of arithmetic operations and experimentally by FPGA implementation. Theoretically speaking, the proposed converter requires one 3:1 adder, three 2:1 adders, and 2 multipliers while the one in [41] requires one 3:1 adder, two 2:1 adders, and four multipliers. In terms of delay the proposed scheme requires 3 or 4 additions and 1 multiplication whereas the converter in [41] requires 3 additions and 2 multiplications. Moreover, due to the fact that our scheme operates on smaller magnitude operands, it utilizes less complex adders and multipliers, which potentially results in even faster and smaller implementations. For the experimental assessment, the converters were described in VHDL and implemented on Xilinx Spartan 3 FPGA. We used a wide range of values of n for the implementation. The synthesis results indicate that, on average, the proposed converter is in terms of delay 14% and 15.8% faster with 27% and 21% area decrease, when compared to the reverse converters in [41] and [2], respectively. The proposed converter also consumes, on average, 18% and 8% less power when compared to the converters in [41] and [2], respectively.

- In Chapter 6, we proposed two new novel memoryless residue to binary converters for the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$. First, we simplified the CRT to obtain a reverse converter that requires mod-$(2^{n+1}-1)$ instead of both of mod-$(2^{n+1}-1)$ and mod-(2^n-1) required by state of the art converter. Second, we further reduced the resulting architecture in order to obtain a reverse converter that utilizes only Carry Save Adders (CSAs) and Carry Propagate Adders (CPAs). We reduced the large $(2^{n+1}-1)(2^n-1)$ adder to a simple adder with the consequence of a reduction in the dynamic range from M to M-$(2^n-1)^2$. We demonstrated that the reduction penalty decreases exponentially as n increases and for values of $n \geq 5$, the reduction penalty is negligible ($\leq 0.0078\%$). We resolved the dynamic range limitation problem and proposed another converter, which is valid for the entire dynamic range. Theoretically speaking, the two proposed converters are faster than the most effective equivalent state of the art converter, but one of them requires more hardware resources. Finally, we implemented our

proposals and the most effective equivalent state of the art converter on a Xilinx Spartan 3 FPGA. The obtained results show that, on average and contrary to the theoretical results, our schemes significantly reduced the area cost with no delay penalty.

- In Chapter 7, we proposed two novel high speed memoryless residue to binary converters for the moduli set $\{2^{2n+1}-1, 2^n, 2^n-1\}$. First, we simplified the traditional CRT and obtained a new converter that only requires mod-$(2^{2n+1}-1)$ operations. Further simplifications result in a very simple and efficient hardware structure, composed of only CSAs with End-Around Carries (EACs) and CPAs. We resolved the domain restriction problem and propose another reverse converter, which is valid for the entire dynamic range. The theoretical analysis indicates that the two proposed reverse converters are faster than the one in [88] in terms of delay, but one of them requires more hardware resources. We described the proposed reverse converters and the one in [88] in VHDL and carried out the implementation on an FPGA. We used a wide range of values of n for the implementation. The synthesis results indicate that, on average, the proposed limited range CRT and the proposed entire range CRT, respectively, are capable of performing the reverse conversion 13.40% and 13.20% faster, with extra costs of 3.29% and 22.75% in terms of area when compared with the one in [88]. Additionally, the AT^2 efficiency metric suggests that proposal-I and proposal-II are 24% and 3%, respectively, more efficient than the one in [88].

- In Chapter 8, we proposed three reverse converters for the new moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. Two of the proposed reverse converters, namely CI and CIII are based on the traditional CRT while the third one, namely CII is based on the MRC. The CRT based reverse converters are better in terms of both area and delay, when compared to the MRC based reverse converter. Therefore, it can be inferred that the CRT is a better approach to design reverse converters for this class of moduli sets. The proposed reverse converters are memoryless and adder based and can be easily realized in VLSI circuits. We performed both theoretical and experimental evaluation of our proposal. The theoretical analysis shows the advantages of our scheme, which is supported by the experimental results. The obtained synthesis results suggest that, CI is able to improve the conversion computation time by about 6% while reducing the required hardware area by more than 3% when compared to CIII. On the other hand, CIII improves the conversion speed by about 17% and

reduces the hardware resources by more than 4%, regarding the proposed CII reverse converter. The proposed CIII is about 11% faster and requires about 22% more hardware resources than the converter in [88]. Additionally, the AT^2 metric indicates that the proposed CIII is 5% better than the one in [88].

9.2 Major Contributions

The major contributions of this study can be summarized by the following:

- We introduced an RNS to MRC technique, which addresses the computation of mixed radix digits in such a way that enables the MRC parallelization. The proposed technique maximizes the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. This scheme results in an RNS to MR conversion with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, where k is the number of moduli.

- We generalized a previously proposed technique that was restricted to 5-moduli set such that it can be utilized in conjunction with any RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,k}$. Next, we simplified the computing procedure by maximizing the utilization of the modulo-m_i adders and multipliers present in the RNS processor functional units. This scheme results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(k)$, where k is the number of moduli. .

- We presented an efficient reverse converter for the moduli set $\{2n+2, 2n+1, 2n\}$, which has a common factor of 2. Given that for such a moduli set, CRT cannot be directly applied, we introduced in a formal way a CRT based approach for this case, which requires the conversion of $\{2n+2, 2n+1, 2n\}$ set into a moduli set with relatively prime moduli, i.e., $\left\{\frac{m_1}{2}, m_2, m_3\right\}$, when n is even, $n \geq 2$ and $\left\{m_1, m_2, \frac{m_3}{2}\right\}$, when n is odd, $n \geq 3$. We demonstrated that the moduli set transformation can be easily done and doesn't require the computation of any multiplicative inverses.

- We demonstrated that for the $\{2n+2, 2n+1, 2n\}$ moduli set, the computation of multiplicative inverses can be eliminated and proved that for n even or odd, the hardware realization is the same.

- For the moduli set $\{2n+2, 2n+1, 2n\}$, we presented a low complexity implementation that does not require the explicit use of the modulo operation in the conversion process as it is normally the case in the traditional CRT and other state of the art equivalent converters.

- We provided a general 4-moduli RNS conversion scheme and then presented a compact form of multiplicative inverses for the moduli set $\{2n+3, 2n+2, 2n+1, 2n\}$.

- We introduced a modified CRT that requires mod-m_k instead of mod-M calculations by assuming a general n-moduli set $\{m_i\}_{i=1,k}$, $m_1 > m_2 > m_3 > ... > m_k$, with the dynamic range $M = \prod_{i=1}^{k} m_i$. This scheme can be utilized in conjunction with other well established k-moduli sets.

- We proposed a novel reverse converter for the moduli set $\{2n+1, 2n, 2n-1\}$. We simplified the traditional CRT in order to obtain a reverse converter that utilizes mod-$(2n-1)$ operations instead of mod-$(2n)(2n-1)$ and $(2n+1)(2n-1)$ operations required by the converters in [41] and [2], respectively. Next, we transformed the needed corrective addition of M into a corrective m_3 addition. We then presented a novel low complexity implementation that also doesn't require the explicit use of modulo operation in the conversion process.

- We proposed two novel memoryless residue to binary converters for the $\{2^{n+1}-1, 2^n, 2^n-1\}$ moduli set. We simplified the CRT to obtain a reverse converter that requires mod-$(2^{n+1}-1)$ instead of both of mod-$(2^{n+1}-1)$ and mod-(2^n-1) required by the proposal in [85].

- We proposed two novel high speed memoryless reverse converter for the moduli set $\{2^{2n+1}-1, 2^n, 2^n-1\}$.

- We proposed three memoryless reverse converters for the moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. First, we proposed a novel reverse converter, which is purely adder based, using the traditional CRT. Second, due to the fact that the proposed CRT based structure does not cover the entire dynamic range, two other reverse converters, which cover the entire dynamic range, are proposed.

9.3 Proposed Research Directions

In this thesis, a number of efficient reverse conversion algorithms, based on either the CRT or MRC have been presented. Designing efficient converters is a major step in the realization of RNS based general purpose processors. This section presents some future research that could further improve RNS usage in both general purpose and special purpose processors.

- As discussed in this thesis, one of the ways of designing an efficient MRC based reverse converter is to propose a strategy which can produce unity for all the required multiplicative inverses in the conversion processes. Investigating this will be interesting as the selection of moduli to satisfy the unity criterion is an empirical procedure which needs further attention.

- The difficult RNS operations include division, sign detection, magnitude comparison, and overflow detection. Since the majority of the algorithms for performing these difficult RNS operations are based on reverse conversion, it will be interesting to re-design existing techniques or to design entirely new techniques for handling the difficult RNS operations using the currently available efficient reverse converters.

- Designing an efficient conversion scheme, which has the capability to switch between one moduli set and another during conversion is another possible future work. This will be interesting and is particularly applicable in some cryptographic applications.

- Since performing mod-$(2^n + 1)$ type arithmetic is relatively slower and more complex than performing mod-$(2^n - 1)$ type arithmetic. It will be of interest to design a general reverse converter for the moduli set $\{2^a, 2^b - 1, 2^c - 1\}$, where b is not equal to c.

- The introduction of the New CRT [131] created a lot of possibilities for RNS researchers to develop efficient reverse converters especially for $(2^n - 1, 2^n, 2^n + 1)$ type moduli set. As discussed in Chapter 1 of this thesis, the reverse converter for the moduli set $(2^n - 1, 2^n, 2^n + 1)$ has been generalized in [52] and requires only one carry save adder and a carry propagate adder, which is very efficient. Given that larger dynamic range is of practical interest, providing a generalised efficient reverse converter structure for the moduli set $\{2^a, 2^b - 1, 2^b + 1, 2^{2b} + 1\}$ will be of interest.

- In literature, the existing RNS fault tolerant architectures are based on either the traditional MRC or CRT. Now, efficient reverse conversion algorithms are available. Thus, designing efficient RNS fault tolerant architectures can be considered as a possible research direction.

- RNS can be used to improve the performance of Smith Waterman's Algorithm (SWA). The SWA is an optimal method for homology searches and sequence alignment in genetic databases and makes all pair wise comparisons between two strings of Deoxyribonucleic Acid (DNA). It achieves high sensitivity as all the matched and near-matched pairs are detected. However, the computation time required strongly limits its use. The SWA involves the basic RNS supported arithmetic operations such as addition and multiplication. Since it has been shown in literature both theoretically and experimentally that using these basic arithmetic operations, RNS is faster than the conventional binary number system, we suggest RNS as an alternative candidate for improving the performance of the SWA. The SWA has been extensively described in [53, 54].

Bibliography

[1] M. Abdallah and A.Skavantzos. On the binary quadratic residue system with non-coprime moduli. *IEEE Transaction on Signal Processing, Vol. 45, No. 8, August*, 1997.

[2] M.O. Ahmad, Y. Wang, and M.N.S. Swamy. Residue to binary number converters for three moduli set. *IEEE Trans. on Circuits and Systems-II, Vol. 46, No.7, pp. 180-183, Feb.*, 1999.

[3] M. Akkal and P. Siy. A new mixed radix conversion algorithm mrc-ii. *Journal of Systems Architecture, vol.53, pp.577-586*, 2007.

[4] E. Al-radadi and P. Siy. A new technique for fast number comparison in the residue number system based on chinese remainder theorem ii. *Proceedings of the MUG 18th International Conference, Denver, October*, 2001.

[5] E. AL-RADADI and P. SIY. Four-moduli set $(2, 2^n - 1, 2^n + 2^{n-1} - 1, 2^{n+1} + 2^n - 1)$ simplifies the residue to binary converters based on CRT II. *PERGAMON Computers and Mathematics with Applications 44, pp. 1581-1587*, 2002.

[6] S. Andraos and H. Ahmad. A new efficient memoryless residue to binary number converter. *IEEE Trans. Circuits Syst., Vol. 35, pp. 1441-1444, Nov.*, 1988.

[7] A.Skavantzos and Y Wang. Implementation issues of the two-level residue number system with pair of conjugate moduli. *IEEE Transactions on Signal Processing, Vol. 47, No.3, March*, 1999.

[8] A.Skavantzos and Y Wang. New efficient RNS-to-weighted decoders for conjugate- pair moduli residue number systems. *Proceedings of ISCAS, pp. 1345-1350*, 1999.

[9] D. K. Banerji. A high-speed division method in residue arithmetic. *IEEE Proc. Fifth Symp. Computer Arithmetic, pp. 158-164, May*, 1981.

[10] A. Z. Baraniecka and G. A. Jullien. Residue number system implementation of number theoretic transforms in complex residue rings. *IEEE Trans. Acoustics, Speech, Signal Processing, vol. ASSP-28, pp. 285-291, June*, 1980.

[11] F. Barsi and P. Maestrini. Error correcting properties of redundant residue number systems. *IEEE Transactions on Computer, Vol. c-22, No. 3 pp.307-315, March*, 1973.

[12] M. Bhardwaj, A.B. Premkumar, and T. Srikanthan. Breaking the 2n-bit carry propagation barrier in residue to binary conversion for the $\{2^n + 1, 2^n, 2^n - 1\}$ moduli set. *IEEE Trans. on Circuits and Syst. II, Vol. 45, pp. 998-1002, September*, 1998.

[13] M. Bhardwaj, T. Srikanthan, and C.T. Clarke. A reverse converter for the 4-moduli superset $\left\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\right\}$. *IEEE Symp. Computer Arithmetic, pp. 168-175, April*, 1999.

[14] G. Bi and E.V. Jones. Fast conversion between binary and residue numbers. *Electronic Letters, vol. 24, no. 19, pp. 1195-1197, September*, 1988.

[15] S. Bi and W.J. Gross. The mixed-radix chinese remainder theorem and its applications to residue comparison. *IEEE Trans. on Computers, vol. 57, No. 12, pp. 1624-1632, December*, 2008.

[16] S. Bi, W. Wang, and A. Al-Khalili. Modulo deflation in $(2^n+1, 2^n, 2^n-1)$ converters. *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'04), Vol. 2, pp.429-432*, 2004.

[17] S. Bi, W. Wang, and A. Al-Khalili. New modulo decomposed residue-to-binary algorithm for general moduli sets. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'04), Vol. 5, pp.141-144*, 2004.

[18] A. D. Booth. A signed binary multiplication technique. *Quaterly J. Math. Appl. Math., Vol. 4, part 2, pp. 236-240*, 1951.

[19] T.K. Callaway and E.E. Swartzlander Jr. Power-delay characteristics of cmos multipliers. *in Proc. 13th Symp. Computer Arithmetic (ARITH13), Asilomar, USA, July 1997, pp. 26-32*, 1997.

[20] B. Cao, C. Chang, and T. Srikanthan. Adder based residue to binary converters for a new balanced 4-moduli set. *Proc. of the 3rd Int. Symp. on Image and Signal Processing Analysis (ISPA03), pp 820-825*, 2003.

[21] B. Cao, C. Chang, and T. Srikanthan. An efficient reverse converter for the 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ based on the new chinese remainder theorem. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSI: FUNDAMENTAL THEORY AND APPLICATIONS, VOL. 50, NO. 10, pp.1296-1303, October*, 2003.

[22] B. Cao, T. Srikanthan, and C. Chang. Design of a high speed reverse converter for a new 4-moduli set residue number system. *Proceedings of the IEEE International Symposium on Circuits and Systems, May 2528, Institute of Electrical and Electronics Engineers Inc., Bangkok, Thailand, pp. 520523*, 2003.

[23] B. Cao, T. Srikanthan, and C. Chang. Design of residue-to-binary converter for a new 5-moduli superset residue number system. *Proceedings of the IEEE International Symposium on Circuits and Systems, 2326 May, IEEE, Vancouver, BC, Canada, pp. 841844*, 2004.

[24] G. C. Cardarilli, M. Er, and R. Lojacona. RNS-to-binary conversion for efficient vlsi implementation. *IEEE Trans. Circuits Syst. II, vol. 45, pp. 667669, June*, 1998.

[25] R. Chaves. Secure computing on reconfigurable systems. *PhD Thesis, Delft University of Technology, Delft, The Netherlands, ISBN 978-90-807957-5-4*, 2007.

[26] R. Chaves and L. Sousa. $\{2^n + 1, 2^{n+k}, 2^n - 1\}$: A new RNS moduli set extension. *Symp. on Digital System Design: Architectures, Methods and Tools, pp. 210-217, September*, 2004.

[27] R. Chaves and L. Sousa. Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures. *IET Comp. Digital Tech., Vol. 5, No.1, pp.472-480, Sept.*, 2007.

[28] T.C. Chen. Maximal redundancy signed-digit systems. *Proceedings of IEEE Symposium on Computer Arithmetic, pp. 296-300, Urbana, June*, 1985.

[29] H. Cohen. A course in computational algebraic number theory. *New York: Springer-Verlag, ISBN 0-387-55640-0*, 1993.

[30] R. Conway and J. Nelson. Fast converter for 3 moduli RNS using new property of crt. *IEEE Trans. Comput., vol. 48, pp. 852-860, August*, 1999.

[31] R. Conway and J. Nelson. Improved RNS fir filter architectures. *IEEE Trans. on Circuits and Systems-II: Express briefs, Vol. 51, No.1, pp. 26-28, January*, 2004.

[32] A. Dhurkadas. An efficient residue-to-decimal converter design. *IEEE Trans. on Circuits and Systems, Vol. 37, pp. 849-850, June*, 1990.

[33] G. Dimauro, S. Impedovo, and G. Pirlo. A new technique for fast number comparison in residue number system. *IEEE Trans. on Computers, vol. 42, No. 5, pp. 608-612, May*, 1993.

[34] P.G. Fernandez, A. Garcia, J. Ramirez, and A. Lloris. Fast RNS-based 2D-DCT computation on field-programmable devices. *Proceedings of the IEEE Signal Processing Systems Workshop, pp.365-373, LA, USA, October*, 2000.

[35] P.G. Fernandez, A. Garcia, J. Ramirez, L. Parrilla, and A. Lloris. A RNS-based matrix-vector-multiply fct architecture for dct computation. *Proceedings of 43rd IEEE Midwest Symposium on Circuits and Systems, pp.350-353, Lansing, MI, August*, 2000.

[36] D. Gallaher, F. Petry, and P. Srinivasan. The digital parallel method for fas RNS to weighted number system conversion for specific moduli $\{2^n + 1, 2^n, 2^n - 1\}$. *IEEE Trans. on Circuits and Systems-II, Vol. 44, pp. 53-57, Jan*, 1997.

[37] H.L. Garner. The residue number system. *IRE Trans. on Electronic Computers, pp. 140-147*, 1959.

[38] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana. High speed rns reverse converters for the moduli set $\left\{2^{2n+1} - 1, 2^n, 2^n - 1\right\}$. *IEEE Trans. on Circuits and Systems II (to appear)*, 2010.

[39] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana. An improved reverse converter for the $\left\{2^{2n+1} - 1, 2^n, 2^n - 1\right\}$ moduli set. *IEEE International Symposium on Circuits and Systems (ISCAS 2010) , Paris, France, June*, 2010.

[40] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana. Modified rns to binary converters for the moduli set $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. *Book chapter of the 2nd IEEE Int. Conference on Adaptive Science and Technology (ICAST'09) (to appear)*, 2010.

[41] K.A. Gbolagade and S.D. Cotofana. An efficient RNS to binary converter using the moduli set $\{2n+1, 2n, 2n-1\}$. *XXIII conference on Design of Circuits and Integrated Systems (DCIS 2008),Grenoble, France, November*, 2008.

[42] K.A. Gbolagade and S.D. Cotofana. Generalized matrix method for efficient residue to decimal conversion. *Proceedings of 10th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2008), pp. 1414-1417, Macao,China, December*, 2008.

[43] K.A. Gbolagade and S.D. Cotofana. MRC technique for rns to decimal conversion for the moduli set $\{2n+2, 2n+1, 2n\}$. *Proceedings of 16th Annual Workshop on Circuits, Systems, and Signal processing, pp. 318-321, Veldhoven, The Netherlands, 27-28, November*, 2008.

[44] K.A. Gbolagade and S.D. Cotofana. Residue number system operands to decimal conversion for 3-moduli sets. *Proceedings of 51st IEEE Midwest Symposium on Circuits and Systems, pp.791-794, Knoxville, USA, August*, 2008.

[45] K.A. Gbolagade and S.D. Cotofana. A residue to binary converter for the $\{2n+2, 2n+1, 2n\}$ moduli set. *Proceedings of 42nd Asilomar Conference on Signals, Systems, and Computers, pp. 1785-1789, California, USA, October*, 2008.

[46] K.A. Gbolagade and S.D. Cotofana. An o(n) residue number system to mixed radix conversion. *Proceedings of the 2009 IEEE International Symposium on Circuits and Systems (ISCAS 2009), pp. 521-524, Taiwan, China, May*, 2009.

[47] K.A. Gbolagade and S.D. Cotofana. Residue-to-decimal converters for moduli sets with common factors. *Proceedings of 52st IEEE Midwest Symposium on Circuits and Systems, pp. 624-627, Cancun, Mexico, August*, 2009.

[48] K.A. Gbolagade and S.D. Cotofana. A reverse converter for the new 4-moduli set $\{2n+3, 2n+2, 2n+1, 2n\}$. *Proceedings of the 2009*

IEEE International Conference on Electronics Circuits and Systems (ICECS09), pp. 113-116, Hammamet, Tunisia, December, 2009.

[49] K.A. Gbolagade, G.R. Voicu, and S.D. Cotofana. An efficient fpga design of residue-to-decimal converter for the moduli set $\{2n+1, 2n, 2n-1\}$. *IEEE Trans. on Very Large Scale Integration (in press)*, 2010.

[50] K.A. Gbolagade, G.R. Voicu, and S.D. Cotofana. An efficient fpga design of reverse converter for the moduli set $\{2n+2, 2n-1, 2n\}$. *5th International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2010), pp. 117-120, Terrassa, Spain, July*, 2010.

[51] V.T. Goh and M.U. Sidiqqi. Multiple error detection and correction based on redundant residue number systems. *IEEE Trans. on Communications, Vol. 56, No. 3, pp. 325-330*, 2008.

[52] A. Hariria, K. Navi, and R. Rastegar. A new high dynamic range moduli set with efficient reverse converter. *Computers and Mathematics with Applications vol. 55, pp. 660668*, 2008.

[53] L. Hassan and Z. Al-Ars. Performance improvement of the smith - waterman algorithm. *Annual workshop on circuits, systems and signal processing (ProRISC 2007). Veldhoven, The Netherlands, November*, 2007.

[54] L. Hassan, Z. Al-Ars, and S. Hamdioui. Hardware acceleration of sequence alignment algorithms - an overview. *Proceedings of International conference on Design and Technology of Integrated Systems in Nano cell Era (DTIS 2007), pp 96 - 101, Rabat, Morocco*, 2007.

[55] A. A. Hiasat. Vlsi implementation of new arithmetic residue to binary decoders. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 13, No.1, pp. 153-158, January*, 2005.

[56] A.A. Hiasat. Efficient residue to binary converter. *IEEE Proceedings Digital Tech*, 2003.

[57] A.A. Hiasat and H.S. Abdel-AtyZohdy. Residue-to-binary arithmetic converter for the moduli set $\{2^k, 2^k-1, 2^{k-1}-1\}$. *IEEE Transactions on Circuits and Systems-II Analog and Digital Signal Processing, Vol.45, No. 2, pp. 204-209, Feb.*, 1998.

[58] C. H. Huang and F. J. Taylor. High speed DFTs using residue numbers. *in Proc. IEEE 1980 Conf Acoust., Speech, Signal Processing, Denver, Co, pp. 238-241, April*, 1980.

[59] C.H. Huang. A fully parallel mixed-radix conversion algorithm for residue number applications. *IEEE Trans. Computers, Vol. C-32, pp. 398-402, April*, 1983.

[60] K. Ibrahim and S. Saloum. An efficient residue to binary converter design. *IEEE Trans. on Circuits and Systems, Vol. 35, pp 1156-1158, Sep.*, 1988.

[61] M.K. Ibrahim. Novel digital filter implementations using hybrid rns-binary arithmetic. *Signal Processing, vol. 40, no. 2-3, pp. 287-294*, 1994.

[62] ITRS. International technology roadmap for semiconductors, emerging research devices. *ITRS report (Executive Summary), http://www.itrs.net/Common/2007ITRS*, 2007.

[63] G. Jaberipur and S. Gorgin. An improved maximally redundant signed digit adder. *Journal of Computers and Electrical Engineering, Vol. 36, pp. 491-502*, 2010.

[64] W.J. Jenkins. Techniques for residue-to-analog conversion for residue-encoded digital filters. *IEEE Trans. on Circuits and Syst., Vol. CAS-25, pp. 555-562, July*, 1978.

[65] W.K. Jenkins and B.J. Leon. The use of residue number systems in the design of finite impulse response digital filters. *IEEE Trans. on circuit and systems, vol. 24, pp. 191-200*, 1977.

[66] W.A. Chren Jr. One-hot residue coding for low delay-power product cmos design. *IEEE Trans. Circuits Syst. II, vol. 45, pp. 303-313, March*, 1998.

[67] G.A. Jullien. Residue number scaling and other operations using ROM arrays. *IEEE Trans. Computers, Vol. C-27, pp. 325-336, April*, 1978.

[68] G.A. Jullien, W.C. Miller, J.J. Soltis, A. Baraniecka, and B. Tseng. Hardware realization of digital signal processing elembnts using the residue number system. *IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Hartford, CT,vol. 2, pp. 506-510, May*, 1977.

[69] K.D.Tocher. Technique for multiplication and division for automatic binary computers. *Quaterly J. Math. Appl. Math., Vol. 11, part 3, pp. 364-384*, 1958.

[70] D.E. KnuthR. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. 3rd ed., Addison-Wesley, 1997.

[71] C. K. Koc. An improved algorithm for mixed radix conversion of residue numbers. *Journal of Comp. Math. App., Vol. 22, No. 8, pp. 63-71, Great Britain*, 1991.

[72] Yuan-Ching Kuo, Su-Hon Lin, Ming-Hwa Sheu, Jia-You Wu, and Peng-Siang Wang. Efficient vlsi design of a reverse RNS converter for new flexible 4-moduli set $(2^{p+k}, 2^p+1, 2^p-1, 2^{2p}+1)$. *In proceedings of the 2009 IEEE International Symposium on Circuits and Systems (ISCAS 2009), pp. 437-440, Taiwan, China, May*, 2009.

[73] P.E. Landman and J.M. Rabaey. Architectural power analysis: The dual bit type method. *IEEE Trans. VLSI Syst., vol. 3, pp. 173-187, June*, 1995.

[74] K. Y. Lin, B. Krishna, and H. Krishna. Rings, fields, the chinese remainder theorem and an extension. *IEEE Trans. Circuits Syst. II , vol. 41, pp. 641655, October*, 1994.

[75] S. Lin, M. Sheu, and C. Wang. Efficient VLSI design of residue to binary converter for the moduli set $\{2^n, 2^{n+1}-1, 2^n-1\}$. *IEICE Trans. INF. and SYST., Vol. E91-D, No.7, pp. 2058-2060, July*, 2008.

[76] M. Lu and J. Chiang. A novel division algorithm for the residue number system. *IEEE Trans. Computers, (special issue on comp. arithmetic) Vol. 41, issue 8, pp. 1026-1032, August*, 1992.

[77] Mi Luu. Arithmetic and logic in computer systems. *John Wiley and Sons, New Jersey*, 2004.

[78] A.S. Madhukumar and F. Chin. Performance of a residue number system based cdma system over bursty communication channels. *Journal of Wireless Personal Communications, Springer, Netherlands, Vol. 22, No. 1, pp.89-102*, 2002.

[79] A.S. Madhukumar, F. Chin, and A.B. Premkumar. Residue number system based multicarrier CDMA for broadband mobile communication

systems. *Proceedings of 43rd IEEE Midwest Symposium on Circuits and Systems, pp.536-539, Lansing, MI, August*, 2000.

[80] R.D. Merrill. Improving digital computer performance using residue number theory. *Trans. on Electronic Computers, vol. 13, issue 2, pp. 93-101*, 1964.

[81] D.F. Miller and W.S. McCormick. An arithmetic free parallel mixed radix conversion algorithm. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 45, No.1, pp. 158-162, January*, 1998.

[82] P. V. Ananda Mohan and A. B. Premkumar. RNS-to-binary converters for two four-moduli sets $\{2^n, 2^n + 1, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^n, 2^n + 1, 2^n - 1, 2^{n+1} + 1\}$. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSI: REGULAR PAPERS, VOL. 54, NO. 6, pp.1045-1054, June*, 2007.

[83] P.V. Ananda Mohan. Residue number systems algorithms and architectures. *The Kluwer Int. Series in Eng. Ans Science, the Netherlands*, 2002.

[84] P.V.A. Mohan. Reverse converters for the moduli sets $2^{2n} - 1, 2^n, 2^{2n} + 1$ and $2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3$. *Proceedings of International Conference on Signal Processing and Communications (SPCOM), pp.188-192*, 2004.

[85] P.V.A. Mohan. Rns-to-binary converter for a new three-moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$. *IEEE Trans. on Circuits and Systems-II: Express briefs, Vol. 54, No.9, pp. 775-779, September*, 2007.

[86] A.S. Molahosseini and K. Navi. New arithmetic residue to binary converters. *International Journal of Computer Sciences and Engineering Systems, Vol. 1, No.4, pp. 295-299, October*, 2007.

[87] A.S. Molahosseini, K. Navi, O. Hashemipour, and A. Jalali. An efficient architecture for designing reverse converters based on a general three-moduli set. *Journal of Systems Architecture, Vol. 54, pp. 929-934*, 2008.

[88] A.S. Molahosseini, K. Navi, and M.K. Rafsanjani. A New residue to binary converter based on mixed-radix conversion. *3rd International Conference On Information and Communication Technologies: From Theory to Applications (ICTTA 2008), pp. 1-6, April*, 2008.

[89] J.S. Soundararajan N.B. Chakraborti and A.L.N. Reddy. An implementation of mixed-radix conversion for residue number applications. *IEEE Trans. Computers, Vol. C-35, Aug.*, 1986.

[90] H. Nussbaumer. Digital filtering using cotransforms in finite fields. *Electronic Letters, Vol. 12, No. 5, pp. 113-114*, 1976.

[91] V. Paliouras and T. Stouraitis. Low-power properties of the logarithmic number system. *in Proc. 15th Symp. Computer Arithmetic (ARITH15)*, 2001.

[92] V. Paliouras and T. Stouraitis. Signal activity and power consumption reduction using the logarithmic number system. *in Proc. 2001 IEEE Int. Symp. Circuits and Systems (ISCAS), vol. 2, pp. 653-656*, 2001.

[93] B. Parhami. Generalized signed-digit number system: a unifying framework for redundant number representation. *IEEE Transactions on Computers, Vol. 39, No. 1, pp. 89-98*, 1990.

[94] B. Parhami. Computer arithmetics:algorithms and hardware designs. *New York, Oxford University Press*, 2000.

[95] K.K. Parhi. Low-energy csmt carry generators and binary adders. *IEEE Trans. VLSI Syst., vol. 7, pp. 450-462, December*, 1999.

[96] S. Piestrak. A high-speed realization of a residue to binary number system converter. *IEEE Trans. on Circuits and Systems-II, Vol. 42, No. 10, pp 661-663, Oct.*, 1995.

[97] F. Pourbigharaz and H. M. Yassine. A signed-digit architecture for residue to binary transformation. *IEEE Trans. Comput., vol. 46, pp. 11461150, October*, 1997.

[98] A. B. Premkuma. Corrections to an RNS to binary converter in a three moduli set with common factors. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Processing, Vol. 51, No.1, pp 43, January*, 2004.

[99] A.B. Premkumar. An RNS to binary converter in $\{2n+1, 2n, 2n-1\}$ moduli set. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 39, No.7, pp. 480-482, July*, 1992.

[100] A.B. Premkumar. Residue to binary converter in three moduli set with common factors. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 42, No.4, pp. 298-301, April*, 1995.

[101] A.B. Premkumar. A formal framework for conversion from binary to residue numbers. *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 46, no. 2, pp. 135144, February*, 2002.

[102] A.B. Premkumar, E.C. Ang, and E.M.K. Lai. Improved memoryless rns forward converter based on the periodicity of residues. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMSII: EXPRESS BRIEFS, VOL. 53, NO. 2, pp.133-137, February*, 2006.

[103] J. E. Robertson. A new class of digital division methods. *IEEE Trans. on Computers, Vol. c-7, pp. 218-222, September*, 1958.

[104] M. R. Schroeder. *Number Theory in Science and Communication*. Germany: Springer-Verlag, 1984.

[105] T.K. Shahana, R.J. Babita, K.P. Jacob, and S. Sasi. RRNS-convolutional concatenated code for OFDM based wireless communication with direct analog-to-residue converter. *Proceedings of World Academy of Science, Engineering and Technology, Vol. 35, pp.652-659*, 2008.

[106] M. Sheu, S. Lin, C. Chen, and S. Yang. An efficient vlsi design for a residue to binary converter for general balance moduli $(2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3)$. *IEEE Trans. Circuits Syst. II, Express briefs, Vol. 51, no. 3, pp.152-155, March*, 2004.

[107] A. Skavantzos and M Abdallah. Novel residue arithmetic processors for high speed digital signal processing. *Proceedings of ISCAS,pages 187-193*, 1998.

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

[109] L. Sousa. Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli. *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, 2007.

[110] T Stouraitis and V. Paliouras. Considering the alternatives in low-power design. *IEEE Circuits and Devices Magazine, Vol. 17, issue 4, pp. 22-29*, 2001.

[111] N. Szabo and R Tanaka. *Residue Arithmetic and its Application to Computer Technology*. MC-Graw-Hill, New York, 1967.

[112] F. Taylor and A.S. Ramnarynan. An efficient residue-to-decimal converter. *IEEE Trans. on Circuits and Systems, Vol. CAS-28, Dec.*, 1981.

[113] F.J. Taylor. Residue arithmetic: A tutorial with examples. *IEEE Computer Society Press, Los Alamitos, CA, USA, Vol. 17, pp.50-62, May*, 1984.

[114] F.J. Taylor. An RNS discrete fourier transform implementation. *IEEE Trans. Acoust. Speech, Signal Process, Vol. 38, No. 8, pp. 1386-1394*, 1990.

[115] F.J. Taylor and C.H. Huang. An autoscale residue multiplier. *IEEE Trans. Computers, Vol. C-31, No. 4, pp. 321-325, April*, 1982.

[116] F.J. Taylor and C.H. Huang. A comparison of DFT algorithms using a residue arithmetic architecture. *International Journal of Computer Electronic Engineering, September*, 1982.

[117] F.J. Taylor, G. Papadourakis, A Skavantzos, and A. Stouraitis. A radix-4 fft using complex rns arithmetic. *IEEE Trans. on Computer, June*, 1985.

[118] T. Toivonen and J. Heikkila. Video filtering with fermat number theoretic transforms based on residue number systems. *IEEE Trans. on Circuits and Systems for Video Tech., Vol. 16, No. 1, pp. 92-101*, 2006.

[119] J. Vaccaro, B. Johnson, and C. Nowacki. A systolic discrete fourier transform using residue number systems over the ring of gaussian integers. *IEEE International Conference on Accoustics, Speech and Signal Processing, pp. 1157-1160*, 1986.

[120] B. Vinnakota and V.V.B. Rao. Fast conversion techniques for binary-residue number systems. *IEEE Trans. on Circuits and Syst. I, Vol. 41, pp. 927-929, Dec.*, 1994.

[121] A.P. Vinod and A.B. Premkumar. A memoryless residue to binary converter for the 4-superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$. *Journal of Circuits, Syst. and Computers, Vol. 10, pp. 85-99,*, 2000.

[122] T.V. Vu. Efficient implementations of the chinese remainder theorem for sign detection and residue decoding. *IEEE Trans. on computers, vol.34, pp.646-651*, 1985.

[123] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang. Applications of the new chinese remainder theorems for three-moduli sets. *in Proc. IEEE Canadian Conf. Electrical and Computer Engineering, vol. 1,pp. 571576, Edmonton, Canada, May*, 1999.

[124] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang. A parallel residue-to-binary converter. *in Proc. IEEE International Conf. Acoustics, Speech, and Signal Processing, Phoenix, AZ, March*, 1999.

[125] W. Wang, M.N.S. Swamy, and M.O. Ahmad. An area-time efficient residue-to-binary converter. *Proceedings of 43rd IEEE MidWest Symp. On Circuits and Systems, Lansing MI, Aug.8-11*, 2000.

[126] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang. A high speed residue-to-binary converter and a scheme for its vlsi implementation. *Proceedings of ISCAS, pp.330-333*, 1999.

[127] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang. A high speed residue-to-binary converter for the three-moduli set $\{2^k, 2^k - 1, 2^{k-1} - 1\}$ rns and a scheme for its VLSI implementation. *IEEE Trans. Circuits Syst. II, Analog Digital Signal Processing, Vol. 10, no. 12, pp.1576-1581, December*, 2000.

[128] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang. A note on a high-speed residue to binary converter for three-moduli $2^k, 2^k - 1, 2^{k-1} - 1$ RNS and a scheme for its VLSI implementation. *IEEE Trans. Circuits Syst. II. Analog Digital Signal Processing, Vol. 49, no. 3, pp.230, March*, 2002.

[129] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang. A study of the residue-to-binary converters for the three-moduli sets. *IEEE Trans on Circuits and Syst.-I, Vol. 50, No.2, pp. 235-243, Feb.*, 2003.

[130] Y. Wang. New chinese remainder theorems. *in Proc. Asilomar Conference, USA, pp. 165-171, Nov.*, 1998.

[131] Y. Wang. Residue-to-binary converters based on new chinese remainder theorems. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 47, No.3, pp. 197-205, March*, 2000.

[132] Y. Wang, X. Song, and M. Aboulhamid. A new algorithm for rns magnitude comparison based on chinese remainder theorem ii. *IEEE Conference, Ninth Great Lakes Symposium on VLSI, 1999, Proceedings, 46, pp. 362365, March*, 1999.

[133] Y. Wang, X. Song, M. Aboulhamid, and H. Shen. Adder based residue to binary number converters for $\{2^n + 1, 2^n, 2^n - 1\}$. *IEEE Trans. on Signal Processing, Vol. 50, pp.1772-1779, July*, 2002.

[134] R.W. Watson and C.W. Hastings. *Residue Arithmetic and Reliable Computer Design*. Washington DC, Spartan, 1967.

[135] From WIKIPEDIA. http://en.wikipedia.org/wiki/$euclidean_algorithm$.

[136] W.Jenkins and B. Leon. The use of residue number systems in the design of finite impulse response digital filters. *IEEE Transaction on Circuits and Systems, Vol.24, pp. 191-201, April*, 1987.

[137] L. Yang and L. Hanzo. Redundant residue number system based error correction codes. *IEEE Vehicular Technology Conference, Vol. 3, pp.1472-1476, Atlantic, NJ, USA*, 2001.

[138] L.L. Yang and L. Hanzo. Residue number system assisted fast frequency-hopped synchronous ultra-wideband spread-spectrum multiple-access: A design alternative to impulse radio. *IEEE Journal on Selected Areas of Communications, 20 (9), pp. 1652-1663*, 2002.

[139] H.M. Yassine. Matrix mixed-radix conversion for rns arithmetic architectures. *34th Midwest Symposium on Circuits and Systems, pages 273-278*, 1992.

[140] H.M. Yassine. Fast arithmetic based on residue number system architectures. *IEEE ISCAS '99, pages 2947-2950, Singapore*, 1999.

[141] H.M. Yassine and W.R. Moore. Improved mixed-radix conversion for residue number architectures. *IEEE Proceedings, vol. 24, pp. 191-200, February*, 1991.

[142] W. Zhang and P. Siy. An efficient design of residue to binary converter for four moduli set $(2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3)$. *ScienceDirect Information Sciences, 178, pp. 264-279*, 2008.

[143] R. Zimmermann. Binary Adder Architectures for cell-based VLSI and their Synthesis. *PhD Dissertation, Swiss Federal Inst. of Technology*, 1997.

List of Publications

International Journals

1. K.A. Gbolagade, G.R. Voicu, and S.D. Cotofana, **"An Efficient FPGA Design of Residue-to-Decimal Converter for the Moduli Set"** $\{2n+1, 2n, 2n-1\}$, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2010 (in press).*

International Conference Proceedings

1. K.A. Gbolagade, G.R. Voicu, and S.D. Cotofana, **"A Memoryless RNS-to-Binary Converter for the** $\{2^{n+1}-1, 2^n, 2^n-1\}$ **Moduli Set"**, *21st IEEE International Conference on Application Specific Systems, Architectures, and Processors (ASAP 2010), pp. 301-304, Rennes, France, July, 2010.*

2. K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana, **"An Improved RNS 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, Paris, France, June, 2010.*

3. K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana, **"Residue-to-Binary Converters for the** $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$ **Moduli Set"**, *2nd IEEE International Conference on Adaptive Science and Technology, pp. 26-33, Accra, Ghana, December, 2009.*

4. K.A. Gbolagade and S.D. Cotofana, **"A Reverse Converter for the New 4-Moduli Set** $\{2n+3, 2n+2, 2n+1, 2n\}$**"**, *16th IEEE International Conference on Electronics, Circuits and Systems, pp. 113-116, Tunisia, December, 2009.*

5. K.A. Gbolagade and S.D. Cotofana, **"Residue-to-Decimal Converters for Moduli Sets with Common Factors"**, *52nd IEEE International*

Midwest Symposium on Circuits and Systems, (MWSCAS 2009), pp. 624-627, Cancun, Mexico, August, 2009.

6. K.A. Gbolagade and S.D. Cotofana, **"An O(n) Residue Number System to Mixed Radix Technique"**, *IEEE International Symposium on Circuits and Systems (ISCAS 2009), pp. 521-524, Taipei, Taiwan, China, May, 2009.*

7. K.A. Gbolagade and S.D. Cotofana, **"Generalized Matrix Method for Efficient Residue to Decimal Conversion"**, *10th IEEE Asia Pacific Conference on Circuits and Systems,pp. 1414-1417, Macao, China, December, 2008.*

8. K.A. Gbolagade and S.D. Cotofana, **"An Efficient RNS to Binary Converter Using the Moduli Set** $\{2n+1, 2n, 2n-1\}$**"**, *XXIII Conference on Design of Circuits and Integrated Systems (DCIS 2008), ISBN 978-2-84813-124-5, Greenoble, France, November, 2008.*

9. K.A. Gbolagade and S.D. Cotofana, **"A Residue to Binary Converter for the** $\{2n+1, 2n, 2n-1\}$ **Moduli Set"**, *Asilomar Conference on Signals, Systems, and Computers, pp. 1785-1789, California, USA, October, 2008.*

10. K.A. Gbolagade and S.D. Cotofana, **"Residue Number System Operands to Decimal Conversion for 3-Moduli Sets"**, *51st IEEE International Midwest Symposium on Circuits and Systems, (MWSCAS 2008), pp. 791-794, Knoxville, U.S.A., August, 2008.*

Local Conference/Workshop Proceedings

1. K.A. Gbolagade and S.D. Cotofana, **"MRC Technique for RNS to Decimal Conversion for the Moduli Set** $\{2n+2, 2n+1, 2n\}$**"**, *16th Annual Workshop on Circuits, Systems, and Signal processing, pp. 318-321, Veldhoven, The Netherlands, November, 2008.*

2. K.A. Gbolagade, G.R. Voicu, and S.D. Cotofana, **"An Efficient FPGA Design of Reverse Converter for the Moduli Set** $\{2n+2, 2n+1, 2n\}$**"**, *5th International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES 2010), pp. 117-120, Terrassa, Spain, July 11-17, 2010.*

Other publication, not directly related to this dissertation:

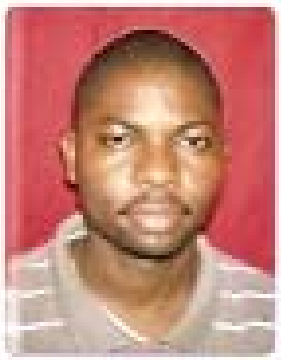
1. Baagyere, E.Y., Boateng, K.O., K.A. Gbolagade, **Bioinformatics: An Important Application Area of Residue Number System**, *in Journal of Communication and Computer, USA, 2009.*

Samenvatting

In deze proefschrift, stellen wij effectieve conversie technieken van RNS naar WNS voor. Dit onderzoek volgt de twee traditionele conversie methoden: de Mixed Radix Conversion (MRC) en de Chinese Remainder Theorem (CRT). In het eerste deel van de research onderzoeken we twee op MRC gebaseerde technieken met k de number of moduli. Als eerst introduceren we een RNS naar MRC techniek, die ingaat op de berekening van Mixed Radix Digits op een zodanige wijze dat deze MRC parallelisatie mogelijk maakt. Dit concept resulteert in een RNS naar MRC met een asymptotische complexiteit, in termen van rekenkundige bewerkingen, in de order van $O(k)$. Ten tweede generalizeren we een eerder voorgestelde techniek die gelimiteerd was tot de 5-moduli verzameling, zodanig dat deze gebruikt kan worden in combinatie met elke RNS voor de verzameling van relatieve priemgetallen moduli $\{m_1, m_2, m_3, ..., m_k\}$. Net zoals voor het eerste concept, resulteert deze tweede techniek ook in een RNS naar MRC met een asymptotische complexiteit, in termen van wiskundige bewerkingen in de order van $O(k)$. In het tweede deel van de research stellen wij een aantal efficiente converters gebaseerd op de simplificatie van de traditionele CRT voor. Ten eerste, nemen we een algemene $\{m_1, m_2, m_3, ..., m_k\}$ moduli verzameling aan, waar $m_1 > m_2 > m_3 > ... > m_k$, met het dynamische bereik $M = \prod_{i=1}^{k} m_i$ en introduceren we een gemodificeerde CRT die mod-m_k in plaats van mod-M berekeningen vereist. Vervolgens, vereenvoudigen we de conversie proces verder door te focussen op moduli verzamelingen met gemeenschappelijke factoren, d.w.z. $\{2n+2, 2n+1, 2n\}$ en $\{2n+3, 2n+2, 2n+1, 2n\}$. Daarnaast, voor de moduli verzameling $\{2n+2, 2n+1, 2n\}$, stellen wij verdere simplificaties voor die in een concept resulteert die zelfs geen explicite modulo operatie bewerkingen vereist. Ten tweede, voor de $\{2n+1, 2n, 2n-1\}$ moduli verzameling, stellen we een nieuwe converer voor, die ook geen explicite modulo operatie bewerkingen vereist gedurende de conversie processen. Ten derde, stellen we twee efficiente geheugen vrije reverse converters voor, voor de $\{2^{n+1}-1, 2^n, 2^n-1\}$ moduli verzameling. Ten vierde stellen we voor de moduli verzameling $\{2^{2n+1}-1, 2^n, 2^n-1\}$ twee efficinte teller gebaseerde converters voor. Tot slot, twee CRT en n MRC gebaseerde reverse converters worden voorgesteld voor de moduli verzamling $\{2^{2n+1}-1, 2^{2n}, 2^n-1\}$. Experimentele resultaten duiden aan dat onze voorstellen equivalente converters in de huidige stand van de techniek substantieel overtreffen in termen van

oppervlakte, berekeningstijd en vermogensverbruik.

Curriculum Vitae



Kazeem Gbolagade was born in Iwo (Osun State), Nigeria, on the 27th of August, 1974. He received his B.Sc degree in 2000 in Computer Science from the University of Ilorin, Kwara State, Nigeria. In 2004, he obtained his Masters degree from the University of Ibadan, Nigeria. In parallel, he worked as a Graduate Assistant at the Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria. He was upgraded to Assistant Lectureship position on the 2nd of September, 2004. On the 12th of September, 2005, he was promoted to Lecturer II. He joined the University for Development Studies, Ghana in 2005 as a Lecturer. He was promoted to Senior Lectureship position in July, 2008 and presently, he is the head of Computer Science Department.

In April 2007, Kazeem Gbolagade joined the Computer Engineering Laboratory group at the Delft University of Technology (TU Delft), The Netherlands. In TU Delft, he pursued a PhD degree under the supervision of Prof. Sorin Cotofana. During his PhD studies, Kazeem worked on Effective Data Conversion in Residue Number System Processors.

From April 26th to June 26th, 2009, Kazeem Gbolagade was a visiting researcher at the Technical University of Lisbon (TU Lisbon) in Portugal. He is a member of the IEEE. During his PhD studies, he participated in the teaching of computer arithmetic (M.Sc class) and served as a peer reviewer in several major international journals and conferences. His research interests include Digital Logic Design, Computer Arithmetic, Residue Number Systems, VLSI Design, and Numerical Computing.