CHAPTER 1
INTRODUCTION

There has been interest in residue number system (RNS) arithmetic as a basis for computational hardware since 1950’s [1]-[3]. An RNS can be thought of as an arithmetic number code that is analogous to the more conventional 1’s-complement and 2’s-complement codes used in commercial digital computers. It is well known that the choice of 1’s-complement or 2’s-complement has a considerable impact on the hardware structure of the machine. Similarly, an RNS code greatly influences the hardware architecture, resulting in a great deal of modularity and parallelism [2].

An RNS is defined by set of integers \( \{m_1, m_2, \cdots, m_n\} \) which are pairwise relatively prime i.e., no pair from the set contains a common non-unity factor. Any natural number \( x \) in the range \([0, M - 1]\), with \( M = \prod_{i=1}^{n} m_i \) may be encoded uniquely by \( n \) residue digits

\[
(x_1, x_2, \cdots, x_n)
\]

where

\[
x_i = |x|_{m_i} \quad \text{for } i = 1, 2, \cdots, n
\]  

(1.1)

Here, \( |x|_{m_i} \) is the non-negative remainder obtained from division of \( x \) by \( m_i \). Residue arithmetic may be defined in general, as

\[
(x_1, x_2, \cdots, x_n) \diamond (y_1, y_2, \cdots, y_n) = (z_1, z_2, \cdots, z_n)
\]  

(1.2)
with \( z_i = |x_i \diamond y_i|_{m_i} \), where \( \diamond \) is one of the operations of addition, subtraction, or multiplication. The property that \( z_i \) depends only on \( x_i \) and \( y_i \) has been referred as carry-free arithmetic, since the processing of \( z_i \) does not interact with the processing of the digits \( z_j \) for \( j \neq i \). Because of this carry-free structure, RNS arithmetic has attracted considerable attention for high-speed computation, especially when high precision is required, i.e., large wordlength is dealt with. In residue arithmetic, all digits are independent which implies that if an error occurs in one digit, it does not propagate into other digit positions during subsequent operations involving addition, subtraction or multiplication. This property provides a basis for fault tolerance. The non-weighted structure of the RNS code is another basic property. These make residue arithmetic useful in the design of fault-tolerant hardware architecture. If a particular residue digit is consistently erroneous, the RNS-based system can still be operated by just isolating the faulty module. Due to carry-free property and fault-tolerant capability, RNS provides parallel arithmetic operations, and it is very much useful in Digital Signal Processing (DSP) applications [4]-[10].

The first published report on the modern work in RNS arithmetic in the context of electronic computers, was that of Svobada [11]. He conducted experiments on a hardwired, small-moduli, RNS computer which he used to study error codes. At approximately the same time Garner [1] presented the basic concepts of RNS arithmetic and suggested its applicability in electronic computers. During the late 1950’s and the early 1960’s, R.A.Baugh and E.C.Day [12], and R.I. Tanaka [13] subsequently developed the theory of RNS arithmetic and attempted to establish its utility for general purpose computers. In 1967, Szabo and Tanaka [2] published a book entirely devoted to the application of RNS arithmetic to modern electronic computers.

By the mid-1960’s, DSP was beginning to emerge as a technical subject distinct from general digital computing. During this era a number of researchers recognized that RNS concepts may offer important advantages for the specialized needs of signal processing,
where the computation is dominated by highly repetitive sequences of multiply and add operations. As for example, digital filtering, digital correlation, fast-fourier transform (FFT) processing, and the calculation of vector matrices require a large number of additions, subtractions, and multiplications. The earliest use of RNS arithmetic in a purely DSP environment was that of Cheney [14], who designed a digital correlator based entirely on residue arithmetic. In early 1970’s, some works on error detection and correction using residue arithmetic were initiated [15]-[17]. During the mid-70’s, the research interest on RNS was significantly focused on the problem of base extension [18], [19]. Here, the RNS representation of an n-moduli system is used to derive the RNS digits for an (n + L) moduli system having their first n moduli in common. Base extension is useful in several fundamental modular arithmetic operations like scaling, magnitude comparison, overflow detection, and sign determination [20]-[23]. During the late 1970’s and 1980’s, many researchers worked on digital filter employing RNS technique [24]-[29]. The advantage of using RNS technique in digital filter is that it consumes less power, and it is also faster as compared to that obtained by using traditional 2’s complement technique. Jenkins and Leon [24] investigated the RNS techniques for non-recursive filters. Soderstrand [25] studied RNS digital ladder structures. Soderstrand and Sinha [29] also discussed about pipelining of RNS based infinite impulse response (IIR) digital filters.

Any RNS based DSP system requires a binary to residue converter at the input end and a residue to binary converter at the output end, while the intermediate stage involves the desired residue based arithmetic unit like residue adder and residue multiplier. During the late 1980’s and the 1990’s many works have been reported on binary to residue converter [30]-[42], residue adder [43]-[57], residue multiplier [58]-[85], residue to binary converter [86]-[123], and RNS based digital filters [124]-[128].

Alia and Martinelli [31] introduced a VLSI computing architecture for conversion of binary number into residue number. For this architecture, a possible layout was given and
its complexity was evaluated in terms of area and time needed for processing. Guan and Jones [33] reported a conversion algorithm for binary to RNS for the moduli set \((2^n - 1, 2^n, 2^n + 1)\) and implemented the converter using binary adders. Pourbigharaz and Yassine [38] presented a fast binary to residue architecture for moduli set \((2^n - 1, 2^n, 2^n + 1)\). They derived a formula to compute the residue of an integer with respect to the modulus \((2^n + 1)\) and replaced the end around carry (EAC) stage with a simple multiplexer. The proposed architecture is free of any modulo adder. Vinnakota and Rao [39] proposed an easy and efficient procedure to convert a binary number into a residue number on moduli set \((2^n - 1, 2^n, 2^n + 1)\). The algorithm is based on inversion technique. They also proposed a procedure to compare the magnitude of two residue number for the above moduli. Ashur et al. [40] proposed a new architecture for fast and efficient conversions from the weighted binary numbers to their three moduli \((2^n - 1, 2^n, 2^n + 1)\) residue representation and vice versa. The converters are realized using carry save arithmetic and modulo adder.

Bayouni et al. [47] proposed three different approaches - the binary adder, the look-up table, and the hybrid approach for realization of residue adder of three different moduli \(m_1 = 2^n\), \(m_2 = 2^n - 1\) and \(m_3 < 2^n - 1\). In binary adder approach, only binary adders are used to implement the residue adder. The look-up table approach is based on storing the RNS functions in look-up tables. In hybrid approach, the residue adder is implemented using both the binary adder approach and the look-up table approach. All three approaches were compared, and the RNS adder was implemented in VLSI medium. Dugdale [51] presented the VLSI implementation of residue adders based on binary adders, which use two cycles of addition and support any class of modulus. Efstathiou et al. [53] discussed an efficient residue adder for modulo \((2^n - 1)\) using the one-level and the two-level carry look-ahead addition algorithms.

Lo and Yang [67] demonstrated a balanced residue number VLSI multiplier which eliminates the extra delay for an unbalanced residue multiplier. Alia and Martinelli [68]
introduced a method to perform the modulo $m$ multiplication which requires only few binary multiplications. Curiger et al. [69] described and compared three new VLSI architectures for multiplication modulo $(2^n + 1)$. In the first approach, they proposed a scheme using an $(n+1) \times (n+1)$-bit multiplier followed by a dedicated modulo correction unit. The second proposal was based on multiplication by modulo $(2^n + 1)$ adders, where the multiplication unit was divided into two parts, a carry save and a final carry-select addition unit. The third approach allowed the application of a bit-pair recoding scheme on the carry-save addition unit, resulting in a significant improvement in both chip area and processing speed. Hiasat [71] presented a memoryless architecture of residue multiplier for moduli $(2^n \pm 1)$, and implemented the multiplier using binary multiplier, adder and discrete logic gates. Radhakrishnan and Yuan [72] proposed two novel approaches for the design of a fast residue number based multiplier over a Galois field $GF(p)$, where $p$ is a prime number. The first approach used an isomorphic mapping from the additive index group, modulo $(p - 1)$, of $GF(p)$ onto the direct sum of a set of sub modular additive groups. The second approach used symmetric residue number arithmetic to perform multiplication. Walter [75] discussed a systolic array for modular multiplication. Dugdale [79] presented a technique for implementation of residue multiplier for non-prime moduli, where the moduli can be decomposed into two or more relatively prime factors. Elleithy and Bayoumi [80] introduced a new algorithm for large moduli RNS multiplication, and designed a systolic architecture to perform the modulo multiplication. Ashur et al. [40] proposed an RNS multiplier for moduli $(2^n \pm 1)$ and implemented the same using binary multiplier, carry save adder and modulo adder. Ma [84] presented a simplified architecture of residue multiplier for modulo $(2^n + 1)$ and used two modulo carry save adders to perform the residue reduction. Zimmermann [57] proposed new VLSI circuit architectures for addition and multiplication modulo $(2^n \pm 1)$ that allow the implementation of highly efficient combinational and pipelined circuits for modular arithmetic.
Residue to binary converters are more complex than binary to residue converters, and researchers are giving efforts to reduce the hardware complexity. Many algorithms have been developed for conversion of residue to binary number. Bernardson [90] described an algorithm and its hardware implementation, which converts the 3 moduli \((2^n - 1, 2^n, 2^n + 1)\) residue numbers into their binary representation. The proposed technique employed only binary adders. Shenoy and Kumaresan [95] presented a new technique of residue to binary conversion using only modular look-up table. Andraos and Ahmad [97] proposed a new architecture of residue to binary converter for moduli \((2^n + 1, 2^n, 2^n - 1)\). Premkumar [108] presented an algorithm and architecture of residue to binary converter for moduli set \((2n + 2, 2n + 1, 2n)\). Wang and Mostafa [111] discussed a divide-and-conquer based RNS decoding algorithm. The basic idea of this algorithm is to decompose a set of moduli into groups, each of size two moduli, and to decode a given number in a two moduli representation. Gallaher *et al.* [112] demonstrated the digit parallel method for conversion of RNS to binary number system of moduli set \((2^n - 1, 2^n, 2^n + 1)\). Pourbigharaz and Yassine [114] presented residue to binary converter architecture based on the Chinese Remainder Theorem (CRT) for general moduli set \(S^k\), where \(S^k = (2^m - 1, 2^{2m} + 1, 2^{3m} + 1, 2^{4m} + 1, \ldots, 2^{km} + 1)\). Hiasat and Zohdy [117] introduced a new algorithm which converts moduli \((2^n, 2^n - 1, 2^{n-1} - 1)\) residue numbers to their binary equivalent, and implemented the converter using adders only. Premkumar [120] proposed new architectures requiring fewer multipliers and adders of smaller size for the conversion of residue number to equivalent binary in the \((2n - 1, 2n, 2n + 1)\) moduli set.

During the 2000’s, research work is mainly focused on residue to binary converter [129]-[142]. However, some papers on binary to residue converters [143],[144], residue adders [145]-[150], residue multipliers [151]-[155], RNS based squarer [156], and finite impulse response (FIR) filters using RNS technique [157]-[160] are also reported in the literatures.
In conventional method of conversion of residue number to binary form, either CRT or Mixed Radix Conversion (MRC) process is used. For residue to binary conversion, the CRT provides a direct formula but it requires extra hardware; on the other hand, the MRC is time consuming as it needs an additional step for conversion from residue to mixed-radix representation. Thus, some efficient conversion algorithms are to be introduced for residue to binary conversion.

Wang [130] introduced a residue to binary converter based on new chinese remainder theorem. Compared to the conventional CRT and the MRC approaches, the new converter requires small size modulo adders. Later, Wang et al. [131] demonstrated an efficient residue to binary converter for moduli set \((2^n+1, 2^n, 2^n-1)\). The efficiency is obtained by an improvement on existing algorithm, and the introduction of a more efficient hardware implementation. Cardarilli et al. [132] presented new architecture for the implementation of a scaled CRT residue decoding algorithm, where the implementation was made using small wordlength look-up tables (LUTs) and simple arithmetic operators. Wang et al. [133] developed a theory for the conversion of a residue number into its binary equivalent for three moduli set \((2^n, 2^n-1, 2^{n-1}-1)\). According to this theory, only one level of modulo operation is needed for the conversion. Wang et al. [135] proposed three new residue to binary converters for moduli set \((2^n-1, 2^n, 2^n+1)\), the proposed circuits consist of either 2n-bit binary adders or n-bit binary adder. Hiasat [136] presented residue to binary converter for five moduli set \((2^{n-2}, 2^n-1, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1)\), where \(n\) is an odd positive integer. Cao et al. [138] described new algorithm of residue to binary conversion for the 4-moduli set \((2^n-1, 2^n, 2^n+1, 2^{n+1}-1)\) where \(n\) is an even number; the four moduli set was treated as the triple moduli set \((2^n-1, 2^n, 2^n+1)\) and a single modulus \((2^{n+1}-1)\). Another algorithm of residue to binary conversion using Quotient Function was reported by Dimauro et al. [139]. This technique is superior to both CRT and MRC for modulus of the kind \(2^n\), where \(n\) is an integer. Sheu et al.
[140] presented an efficient VLSI design for residue to binary converter of four-moduli set \((2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3)\). The merits of the proposed algorithm include: larger dynamic range, higher degree of parallelism for conversion, balanced bit-width for internal RNS arithmetic operations, and flexible moduli set selection. Cao et al. [141] described 5-moduli set \((2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1, 2^{n-1} - 1)\) residue to binary converter with even \(n\). This moduli set provides larger dynamic range and higher parallelism. Their conversion algorithm is based on a 4-moduli set residue to binary conversion and also on the MRC technique. Hiasat and Sweidan [142] proposed a new moduli set \((2^{2n}, 2^n - 1, 2^n + 1)\) residue to binary decoder with a dynamic range of \(4n\)-bits.

Premkumar [143] suggested a new design of binary to residue conversion for arbitrary moduli set which is totally based on combinational circuit, and does not use any memory. Syuto et al. [144] proposed high speed binary to residue converter using Signed-Digit (SD) number representation for moduli \((2^n, 2^n \pm 1)\) without using look-up table.

Hiasat [145] introduced a high speed residue adder for two different class of moduli, \(m < 2^n\) and \(m = (2^n + 1)\) and implemented them using VLSI. Bauchat [148] described three basic methodologies to carry out a modulo \(m\) addition, and also presented a design of modulo \((2^n \pm 1)\) adders. The second part of his work described modulo \(m\) multiplication algorithm involving small multiplier and memory blocks, and modulo \((2^n + 1)\) multipliers based on Ma’s algorithm.

Hiasat [151] presented new residue multiplier for medium and large size moduli. The proposed residue multiplier consists of a \((n \times n)\) binary multiplier, a \(((n - 1 - k) \times k)\) binary multiplier \((k < n)\), three \(n\)-bit adders, and a simple combinational circuit. A combinatorial adder-based RNS multiplier was introduced by Paliouras et al. [153], which is more efficient in the area-time product sense.

Piestrak [156] proposed general design methods of the RNS based squarer circuit for three moduli set \((2^n - 1, 2^n, 2^{n-1} + 1)\), where \(n\) is any integer. The proposed circuits have
been implemented using a network composed of a carry-save adder tree followed by a carry-propagate adder, both with end-around carry.

Mahesh and Mehendale [157] presented algorithmic and architectural transforms for realization of RNS based low power FIR filter. Nannarelli et al. [158] implemented the FIR filter using RNS technique, and compared with filters realized in the traditional two's complement system (TCS) in terms of delay, area and power dissipation. The resulting implementations show that the RNS filters are smaller and consume less power than the corresponding ones in TCS, when the number of taps is larger than sixteen. A new architecture for implementing FIR filter using RNS technique was detailed by Conway and Nelson [160], which is based on using a restricted moduli set, with moduli of the form $2^n$, $2^n - 1$ and $2^n + 1$. The number of moduli is not confined to three, as in the common three moduli set system, but can be any number of relatively prime moduli of this form that satisfies the dynamic range requirements. As compared to 2's complement design, the new design offers a significant speed improvement.

**Scope of the Thesis**

The literature survey reveals that the RNS technique is extremely useful for DSP applications. As already mentioned, quite a large number of algorithms have been developed for binary to residue conversion, residue addition, residue multiplication, and residue to binary conversion. Accordingly, the blocks based on those algorithms form the basis of RNS based DSP system. There is still a wide scope for development of new algorithms for various building blocks of RNS based DSP systems so that hardware complexity is minimum. From the literature it is also clear that the selection of moduli set is an important aspect in RNS based system design. The three moduli set ($2^n - 1$, $2^n$, $2^n + 1$) has the following advantages as compared to other moduli set:
a) Residue addition, subtraction and multiplication with respect to these moduli set are relatively easier than other moduli form.

b) Scaling any RNS with moduli of these form is less complicated.

c) Decoding and encoding of residue digits is relatively simpler than other moduli sets.

Thus, the RNS based on moduli set \((2^n - 1, 2^n, 2^n + 1)\) is expected to play an increasingly important role in DSP systems. As a consequence, lots of papers employ such moduli set [38], [40], [57], [71], [112], [121] etc. The present dissertation also deals with the same moduli set \((2^n - 1, 2^n, 2^n + 1)\).

In this thesis, work is focused on the simulation and implementation of binary to residue converter, residue to binary converter, residue adder and residue multiplier. Simulation works are carried out using TINA (Schematic Editor) which is a powerful software package for designing, simulating and analysing any analog, digital and mixed electronic circuit. Using timing analysis mode of TINA, the time delays between inputs and outputs of the above mentioned building blocks have been measured. Simulation results are verified by implementing the various building blocks using digital ICs and logic gates. All building blocks like binary to residue converter, residue to binary converter, residue adder and residue multiplier are also implemented using 8051 microcontroller kit [161]. The 8051 is an embedded microcontroller and the complete hardware is provided on a single chip [162]. It has also very powerful instruction set and any complex circuit can be implemented very easily. All the RNS based circuits have therefore, been implemented here using 8051 microcontroller. Different flowcharts have been developed for binary to residue converter, residue to binary converter, residue adder and residue multiplier, and on the basis of the flowcharts, programs have been written in Assembly Language. Final outputs have been observed by connecting LEDs at the output port of the microcontroller. VHDL is a hardware description language which is extremely useful for documentation,
verification and synthesis of a digital circuit. Using VHDL code, any digital circuit can be implemented employing CPLD or FPGA. In the present thesis, although hardware realization of various RNS based modules is mainly concerned with microcontroller, following the latest trend of digital circuit implementation, a CPLD based binary to residue converter has also been presented in the later part of the thesis.

This thesis contains 9 chapters including the present one. In chapter 2, basic feature of RNS arithmetic has been described. The chapter also includes the operation of residue addition, residue multiplication, binary to residue conversion and residue to binary conversion with some examples.

Chapter 3 highlights the main features and architecture of 8051 microcontroller. Certain aspects of the architecture (Von-Neumann and Harvard) and memory organization are described with the help of suitable diagrams.

Chapter 4 deals with the algorithms for positive and negative binary to residue conversion of moduli \((2^n - 1, 2^n, 2^n + 1)\) \([163], [164]\). On the basis of algorithms, different circuits of positive and negative binary to residue converters for modulo \((2^n - 1)\) and \((2^n + 1)\) have been presented.

A combinational circuit of residue to binary converter of moduli set \((2^n + 1, 2^n, 2^n - 1)\) has been given in chapter 5 \([165]\).

Residue adders of moduli set \((2^n - 1, 2^n, 2^n + 1)\), and a combined residue adder of moduli \((2^n \pm 1)\) are proposed in chapter 6 \([166]-[169]\).

Chapter 7 deals with RNS multiplier for the same moduli set \([170]\). A combined residue multiplier of moduli \((2^n \pm 1)\) is also presented in this chapter \([171], [172]\).

In chapter 8, the combined circuit of positive and negative binary to residue converter of modulo 15 and modulo 17 has been simulated using VHDL and implemented using CPLD \([173]\).
The thesis concludes with chapter 9, which contains a brief summary of the present investigations and a discussion on the future scope of work along the lines of the investigations reported here.