VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

Ing. DINA YOUNES

RESIDUE NUMBER SYSTEM BASED BUILDING BLOCKS FOR APPLICATIONS IN DIGITAL SIGNAL PROCESSING

VYUŽITÍ SYSTÉMU ZBYTKOVÝCH TŘÍD PRO ZPRACOVÁNÍ DIGITÁLNÍCH SIGNÁLŮ

ZKRÁCENÁ VERZE PH.D. THESIS

ŠKOLITEL: doc. Ing. PAVEL ŠTEFFAN, Ph.D.

BRNO 2013

Keywords

Residue number system, digital signal processing, modular arithmetic, moduli set, dynamic range, binary to RNS converter, RNS to binary converter, RNS-based application, parallel processing, power reduced DSP application, FPGA implementation.

Klíčová slova

Systém zbytkových tříd, digitální zpracování signálu, modulární aritmetika, sada modulů, dynamický rozsah, převodník z binární soustavy do RNS, převodník z RNS do binární soustavy, aplikace RNS, paralelní výpočty, aplikace DSP s nízkou spotřebou, implementace do FPGA.

Table of Contents

1 Introduction

This thesis is concerned with an unconventional non-weighted number system that has gained a great scientific interest; the residue number system (RNS). Designing new and more efficient RNS based building blocks that improve digital signal processing (DSP) applications' performance is the main aim of this thesis.

The RNS is a very old number system. It was found 1500 years ago by a Chinese scholar Sun Tzu. Since the last five decades, RNS's features have been rediscovered and thus the interest in this system has been renewed. The researchers have used the RNS in order to benefit from its features in designing high-speed and fault-tolerance applications.

The fundamental idea of the RNS is based on uniquely representing large binary numbers using a set of smaller residues, which results in carry-free, high-speed and parallel arithmetic. This system is based on modulus operation, where the divider is called modulo and the remainder of the division operation is called residue. The basic notation in the RNS is,

$$x_i = X \bmod m_i = \langle x_i \rangle_{m_i} \quad ; \qquad 0 \le x_i < m_i \tag{1.1}$$

Each integer in RNS is represented by a set of residues corresponding to a specified moduli set. The main condition is that the moduli within the moduli set should be relatively prime,

$$X \xrightarrow{RNS} \left(\langle x_1 \rangle_{m_1}, \langle x_2 \rangle_{m_2}, \cdots, \langle x_n \rangle_{m_n} \right) \quad ; \qquad GCD(m_i, m_j) = 1 \tag{1.2}$$

The RNS uniquely represents any integer X that locates in its dynamic range M, which is the product of the moduli within the moduli set. Any interval of M consecutive integers can be uniquely represented in the RNS.

The principal aspect that distinguishes the RNS from other number systems is that the standard arithmetic operations; addition, subtraction and multiplication are easily implemented, whereas operations such as division, root, comparison, scaling and overflow and sign detection are much more difficult. Therefore, the RNS is extremely useful in applications that require a large number of addition and multiplication, and a minimum number of comparisons, divisions and scaling. In other words, the RNS is preferable in applications in which additions and multiplications are critical. Such applications are DSP, image processing, speech processing, cryptography and transforms [1], [2].

The main RNS advantage is the absence of carry propagation between digits, which results in high-speed arithmetic needed in embedded processors. Another important feature of RNS is the digits independence, so an error in a digit does not propagate to other digits, which

results in no error propagation, hence providing fault-tolerance systems. In addition, the RNS can be very efficient in complex-number arithmetic, because it simplifies and reduces the number of multiplications needed. All these features increase the scientific tendency toward the RNS especially for DSP applications. However, the RNS is still not popular in general-purpose processors, due the aforementioned difficulties.

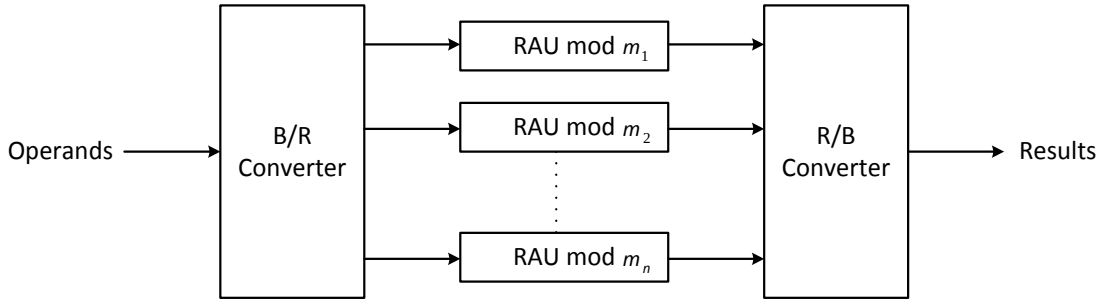


Fig. 1.1: The architecture of the residue number system (RNS)

The basic RNS processor's architecture is shown in Fig. 1.1. It consists of three main components; a forward converter (binary to residue converter), that converts the binary number to *n* equivalent RNS residues, corresponding to the *n* moduli. The *n* residues are then processed using *n* parallel residue arithmetic units (RAU); each of them corresponds to one modulo. The *n* outputs of these units represented in RNS are then converted back into their binary equivalent, by utilizing the reverse converter (residue to binary converter).

1.1 State of the art

The interest in RNS arithmetic has started since 1950's [1], [2]. The first hardware based on the RNS was built in 1967. The work in this field continued and many improvements in all areas of the RNS have arisen, in order to enhance its features, resolve its related problems and find suitable applications that benefit from RNS's features. Most of the early designs of RNS were based on read-only memories (ROM). However, the great advance in VLSI (very large scale integration) technology paved the way for new approaches in designing RNS systems. New trends to design non-ROM based RNS have appeared. Subsequently, much work was devoted for special moduli sets. Excellent results in terms of computational speed have been achieved in 2000 [2].

The most important issues that must be taking into account when designing an RNS system are, a proper moduli set selection, forward conversion, residue arithmetic units and reverse conversion.

2 Aims of the Dissertation

The main objective of this thesis is, *designing, simulation and FPGA implementation of RNS based building blocks for applications in the field of DSP (binary-to-residue converter, residue-to-binary converter, residue adder and residue multiplier).*

Since the RNS results in carry free arithmetic operations and supports high-speed concurrent computations, it will be useful to use RNS-based building blocks for DSP applications.

Therefore, the main objective of this thesis is improving these building blocks by developing new algorithms and improving existing ones. Hence, the aims of this thesis can be categorized as follows,

Studying different moduli sets, analyzing the relationship between the moduli number and the dynamic range it provides and evaluating the most efficient ones for different applications with different dynamic range requirements.

Improving and designing novel RNS converters including both forward and reverse converters. However, the main focus will be concentrated on the reverse converters, since they are the most time and hardware consuming components in the RNS. Comparing ROM-based structures with combinational ones and analyzing the most suitable converters for different applications based on FPGA implementation.

Improving and designing novel structures of residue arithmetic units including modular adders, modular subtractors and modular multipliers with respect to different moduli sets.

Suggesting solutions to simplify RNS difficult operations needed in some DSP applications; such as comparison, overflow and sign detection.

Comparing RNS-based applications with binary-based ones and analyzing the cases when using the RNS will be the most efficient.

Verifying the functionality and efficiency of the proposed designs and comparing them against other published ones based on FPGA implementation.

3 Dissertation Results

This part of the thesis is devoted for presenting the proposed work, findings and results of the doctoral dissertation. In addition to the proposed designs, comparisons of known structures with their analyzing and evaluations are also presented in this Chapter.

3.1 The most efficient moduli set for each dynamic range

Choosing a proper moduli set greatly affects the performance of the whole system. The prevalent issue is that as the number of moduli increases the speed of the residue arithmetic units increases, whereas the residue-to-binary converters become slower and more complex. Thus, I carried out a detailed study on different moduli sets with different moduli numbers and different dynamic ranges and compared timing performance of systems based on them in order to determine the moduli number effect on the overall RNS timing performance and find out the most efficient set for each dynamic range. The study has been published in an international conference in Dubai, UAE [13] and an extended version of it has been published in the international journal of Emerging Trends in Computing and Information Sciences [14].

Based on the analysis and outcomes of this research, the unexpected issue I have ascertained is that, the number of moduli does not affect that much the overall delay of the system considering all its components. Five-moduli sets do not show any superiority over other sets taking into account the three components of RNS (modular adders, modular multipliers and residue to binary converters). Moreover the three-moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$ [4] showed the best timing performance concerning all the three components. Hence, there is no point for choosing a five-moduli set if the overall timing performance will be worse than that based on three or four-moduli sets.

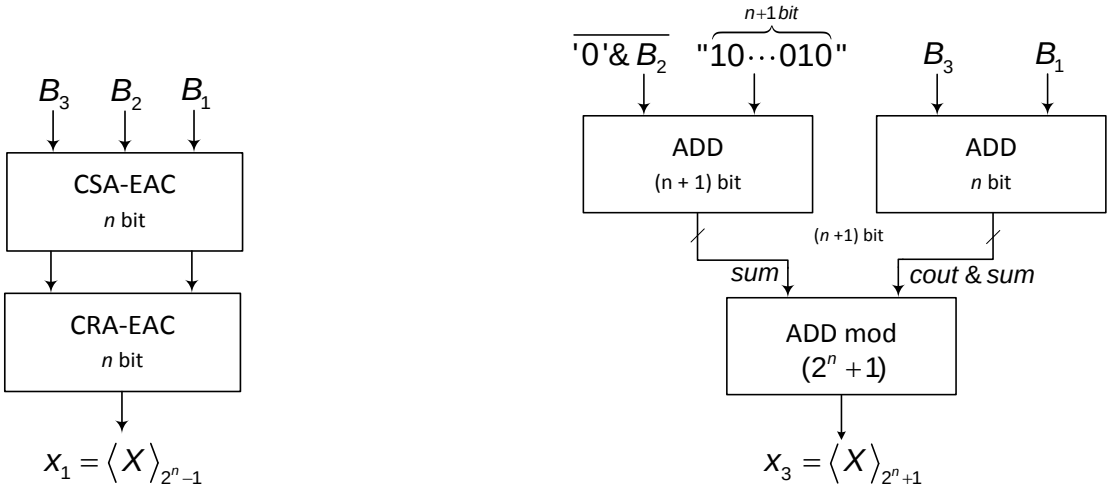
3.2 Proposed forward converter

Due to the fact that binary to residue converters are rather simple, little work has been dedicated to enhance their performance. Since my research dealt with special moduli sets rather than general moduli sets, the utilized components to obtain residues with respect to the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ are presented in this section.

Since the majority of moduli sets have moduli of the following forms $(2^k - 1)$, (2^k) or $(2^k + 1)$, thus, the illustrated forward converters can be used to obtain the RNS representation with respect to any of those sets.

The most straightforward residue to obtain is the one with respect to modulo 2^n. This residue represents the least n bits of the binary number. Thus, no adders or any logical components are needed. However, computing a residue with respect to modulo $(2^n - 1)$,

demands two consecutive modulo $(2^n - 1)$ adders. Instead of using this structure, a carry save adder with end around carry (CSA-EAC) followed by carry ripple adder with end around carry (CRA-EAC) can perfectly fulfill the task. This structure is shown in Fig. 3.1 (a).



(a) (b)

Fig. 3.1: Proposed binary to residue converter – (a) modulo $(2^n - 1)$ channel, (b) modulo $(2^n + 1)$ channel

The most difficult residue to obtain is the one with respect to $(2^n + 1)$ modulo. Typically, this one requires modulo $(2^n + 1)$ subtractor followed by modulo $(2^n + 1)$ adder. This structure is rather complicated, since both components are complex and time consuming.

However, by a proper extraction of the equations needed for the forward conversion process, the proposed structure of the component that computes the residue with respect to modulo $(2^n + 1)$ is considerably simplified. It is realized using two parallel binary adders followed by modulo $(2^n + 1)$ adder as illustrated in Fig. 3.1 (b). Since one of the inputs of the first binary adder is constant, its structure can be simplified, the $(n + 1)$ full adders can be replaced by $(n - 2)$ half adders. However, this simplification does not reduce the delay (due to the second adder that adds $B_1 + B_3$), but the overall hardware complexity decreases.

The proposed forward converter along with pure ROM-based one has been implemented on Virtex-4 XC4VSX25 FPGA. The proposed design was implemented for different dynamic range (DR) requirements (12 bits, 15 bits, 24 bits and 33 bits). Timing performance of the proposed design was very impressive. The maximum frequency of this converter was (353.4 MHz, 292.8 MHz, 275.8 MHz and 231.3 MHz) for (DR = 12 bits, 15 bits, 24 bits and 33 bits), respectively.

A ROM-based converter was also implemented on Virtex-4 FPGA. However, due to the lack of the integrated BRAM count, this converter could only be implemented for two dynamic ranges (12 bits and 15 bits). The maximum frequency of this design was (383.4 MHz

and 258.1 MHz) for (DR = 12 bits and 15 bits), respectively. However, the unexpected issue that has been observed is, that timing performance of the combinational converter for DR = 15 bits is better than the ROM-based one by 13.4%.

Therefore, for large dynamic range requirements, ROM-based converters are not efficient to be implemented (at least on this FPGA device), due to the lack of the integrated BRAM count. Moreover, using external ROMs is not preferable, since they are considerably slower than the built-in ones.

3.3 Proposed residue arithmetic units

The proposed residue arithmetic units including modular adders, modular subtractors and modular multipliers are introduced in this section. All proposed designs can be used with any modulo of the form $(2^k \pm 1)$, hence, they can be used with majority of the published moduli sets. The proposed designs have been published in different national and international conferences and journals [15] – [18].

3.3.1 Proposed modular adders

This section contains the proposed structures of modular adders. Three modular adders were proposed; two of them are specified for modulo $(2^n + 1)$, and one for modulo $(2^n – 1)$. Modulo (2^n) adders have the simplest structures. They can be realized using an *n*-bit binary adder with ignored carry-out. Therefore, my research is focused on modulo $(2^n \pm 1)$ adders.

Modulo $(2^n – 1)$ adder

Majority of the published structures of modulo $(2^n – 1)$ adder perform addition first, and then apply the necessary correction, in order to get the correct result that corresponds to this modulo. The standard structure of this adder depends on two binary adders and a multiplexer. However, the proposed modular adder employs the prefix adders' concept in order to pre-calculate the carry-out needed for the correction process. This design has been published in an international conference in Brno [15] and an extended version has been published in ElectroScope journal [16].

As illustrated in Fig. 3.2, this design contains only one binary adder and a carry-out computation unit, instead of two adders and a multiplexer as stated in [9]. This decreases time and area consumptions in the FPGA.

The proposed adder was compared with an already published design [9], which was denoted as (f). The choice of this adder (f) has been done based on its superiority over other adders stated in [9]. Both adders were implemented on Spartan-3 xc3s200 FPGA. According to the implementation results, the proposed adder has proven its superiority, with savings up to (14.7%, 14.3%) in time, area consumptions, respectively.

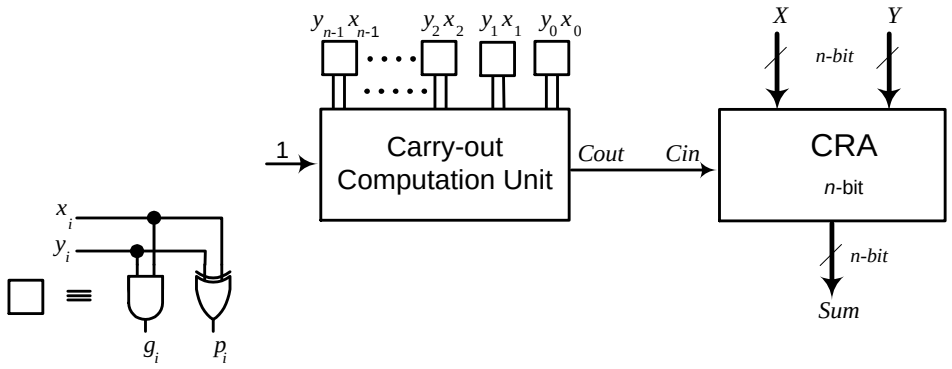


Fig. 3.2: Proposed modulo (2^n – 1) adder – based on prefix carry-out computation [16]

Modulo (2^n + 1) adder

Two different architectures of modulo (2^n + 1) adder were designed. Both adders use normal binary representation instead of diminished-one representation that has two main problems: difficulties in zero representation, and the necessity to converters that convert from/to diminished-one representation. Therefore, I have focused on acquiring the benefits of both representations, i.e. how to speed up the computation process and not face the difficulties in diminished-one representation.

Simple modulo (2^n + 1) adder - by using only n-bit circuits

The structure of this adder is an improved version of that published in an international conference in Brno [17].

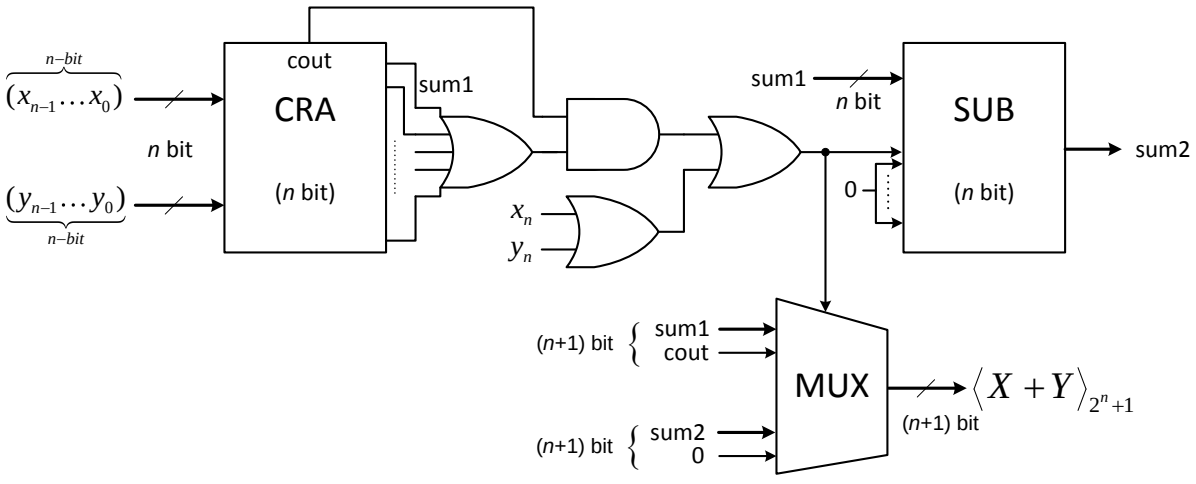


Fig. 3.3: Improved structure of the proposed modulo (2^n + 1) adder – that uses *n*-bit components [17]

The feature of this design is the usage of only *n*-bit circuits instead of (*n* + 1)-bit. In other words, this design uses normal binary representation and at the same time utilizes just *n*-bit circuits, thus, it has the benefits of the two representation methods simultaneously. The

structure of this adder encloses an *n*-bit binary adder, an *n*-bit binary subtractor and a multiplexer. The output is obtained by separately processing the first *n* bits and the MSBs of the operands. The structure of this adder is illustrated in Fig. 3.3.

Modulo (2^n + 1) adder – based on prefix carry computation

Contrary to the previously proposed modulo (2^n + 1) adder, this one consists of (n + 1)-bit circuits. However, it utilizes the concept of prefix carry computation used in parallel prefix adders in order to speed-up the computation process. This modular adder has been published in an international conference in Brno [15] and an extended version has been published in ElectroScope journal [16].

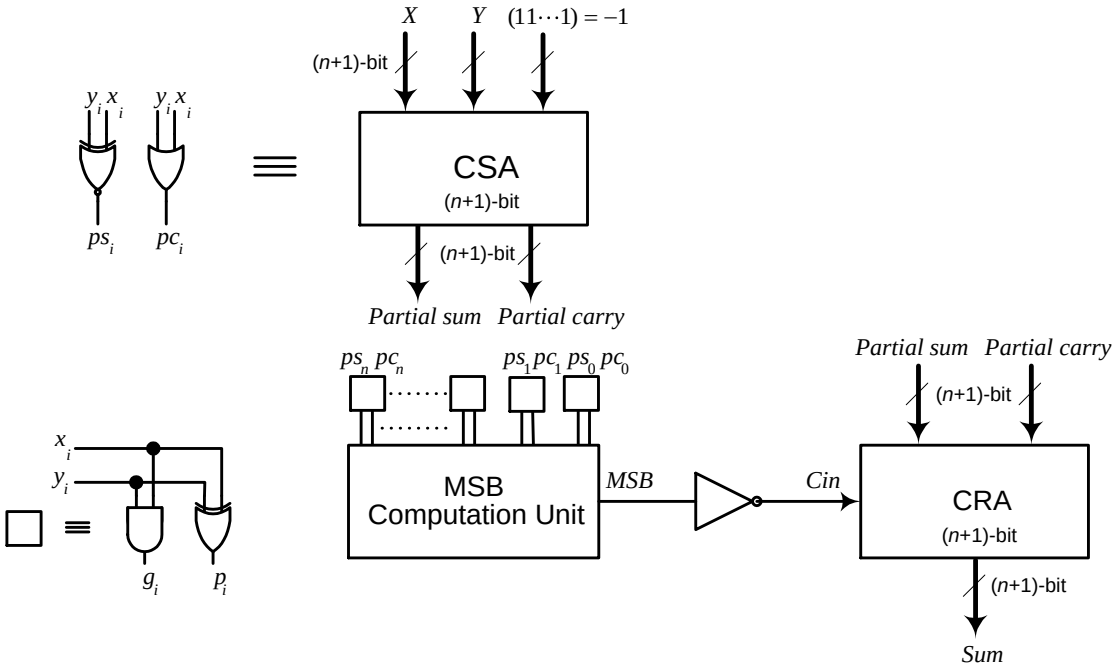


Fig. 3.4: Proposed modulo (2^n + 1) adder - based on the prefix computation [16]

The structure of the proposed adder is illustrated in Fig. 3.4. The main concept of this adder is based on the prefix computation of the MSB of (X + Y – 1), and then applying the necessary correction. This correction is represented in applying the correct carry-in into the CRA.

To prove the efficiency of this adder, it was compared with another already published one, which was published in [9] and denoted as (k). This Modular adder (k) was chosen due to its superiority over other modular adders stated in [9]. Both adders were implemented on Spartan-3 xc3s200 FPGA. The implementation results showed savings up to (37.5%, 13.3%) in time, area consumptions, respectively.

Concerning comparing the two proposed modulo (2^n + 1) adders; the one that uses *n*-bit components [17] with the one based on the prefix computation [16]. The implementation results showed that [17] is superior in terms of area consumption, whereas [16] is superior in terms of time consumption (due to the usage of a CSA that has a delay equal to that of a half adder, and the usage of a prefix carry computation). However, both proposed adders have better performance than that of adder (k) in [9] in terms of time and area consumptions.

3.3.2 Proposed modular subtractor

This section introduces the proposed structure of modulo (2^n + 1) subtractor. Modulo (2^n – 1) subtractor can be simply realized using modulo (2^n – 1) adder and a few inverters. Therefore, only modulo (2^n + 1) subtractor has been proposed. It has been published in an international conference in St. Maarten, the Netherlands Antilles [18]. This subtractor was intentionally designed to be used in the proposed modulo (2^n + 1) multiplier, which has been published in the same paper [18], as will be described later.

The structure of this modular subtractor consists of an (*n* + 1)-bit binary subtractor and an (*n* + 1)-bit binary adder, as shown in Fig. 3.5. The output "*neg*" indicates whether the subtraction result is negative or not. In case of negative result (*neg* = 1), modulo (2^n + 1) should be added back, in order to correct the result. This subtractor has been efficiently utilized in modulo (2^n + 1) multiplier.

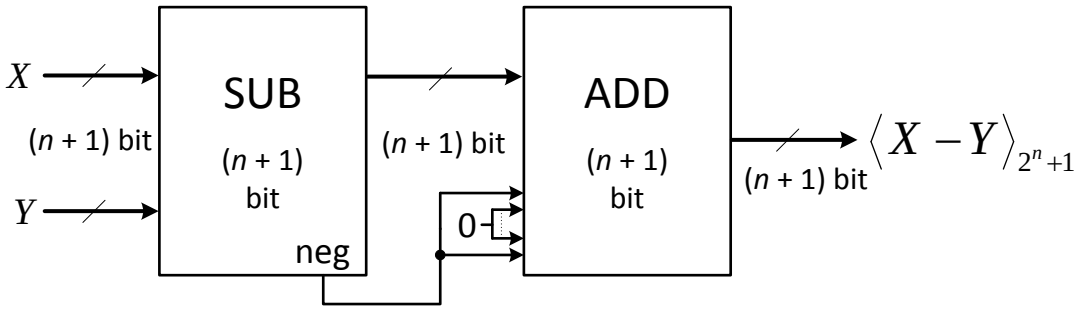


Fig. 3.5: Improved structure of proposed modulo (2^n + 1) subtractor

3.3.3 Proposed modular multipliers

The proposed structures of modulo (2^n ± 1) multipliers are presented in this section. The first part contains a comparison between two structures of modulo (2^n – 1) multiplier. Each of these structures belongs to a different category of modular multipliers. The second part illustrates the proposed modulo (2^n + 1) multiplier that uses the above mentioned modulo (2^n + 1) subtractor as a fundamental component.

Modulo (2^n – 1) multipliers

The first structure of modulo (2^n – 1) multiplier is based on the multiplication-then-reduction method. It is illustrated in Fig. 3.6.

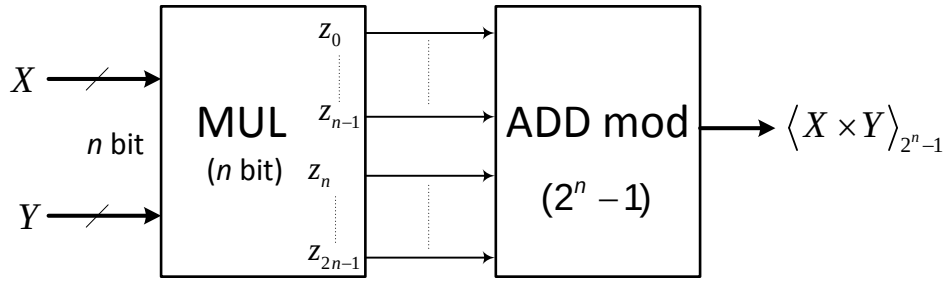


Fig. 3.6: Proposed modulo (2^n – 1) multiplier – based on multiplication-then-reduction approach

However, such a multiplier is quite expensive comparing to the one based on interleaving multiplication and reduction method shown in Fig. 3.7. The structure of this multiplier consists of a rotation unit, a Wallace tree adder to perform multi-operand addition and modulo (2^n – 1) adder.

The second structure has shown better timing performance as the dynamic range increases, whereas, area saving gradually decreases. Therefore, for systems with very large dynamic ranges (more than 60 bits) and whose main goal and strategy is "area reduction", the structure shown in Fig. 3.6 is more effective.

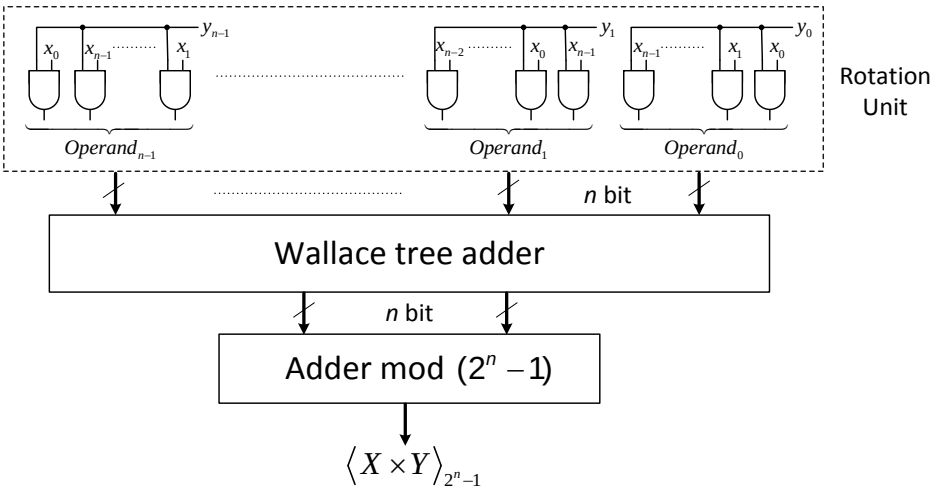


Fig. 3.7: Proposed modulo (2^n – 1) multiplier – based on interleaving multiplication and reduction approach

Modulo (2^n + 1) multiplier

This modulo (2^n + 1) multiplier has been published along with the previous modular subtractor in [18]. This multiplier belongs to the multiplication-then-reduction category. As shown in Fig. 3.8, its structure consists of an (n + 1)-bit binary multiplier followed by modulo (2^n + 1) subtractor. This subtractor performs the reduction process according to modulo (2^n + 1).

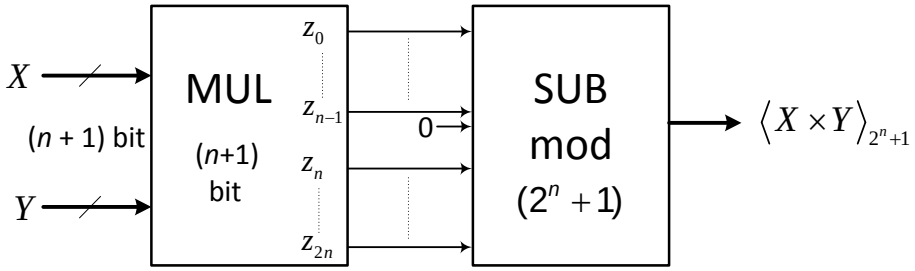


Fig. 3.8: Proposed modulo (2^n + 1) multiplier [18]

The proposed multiplier was compared with an already published modulo (2^n + 1) multiplier [12]. Both multipliers were implemented on Spartan-3 FPGA. The results showed time saving up to 26.8% for medium dynamic range = 12 bits. However, the proposed design shows better timing performance than its counterpart for dynamic range up to 33 bits. The reason is that, for DR $\geq$ 36 bits, the delay of the binary multiplier considerably increases, hence the overall delay of the whole design does. However, for systems with dynamic rages less than 36 bits, the proposed modular multiplier is superior over [12].

3.4 Proposed Reverse converters

This section is dedicated for presenting my work on residue to binary converters. It is divided into two subsections; the first one presents a comparison between two well-known algorithms for residue to binary conversion based on the moduli set {2^n – 1, 2^n, 2^n +1}. The comparison is based on FPGA implementation of these two algorithms. The second part presents a novel algorithm for reverse conversion in the RNS and proposes an efficient reverse converter based on it.

3.4.1 Comparison between the new CRT-I and MRC

This section presents a comparison between two well-known algorithms for residue-to-binary conversion. These algorithms are mixed radix conversion (MRC) and new Chinese remainder theorem (CRT-I). The comparison is done in order to highlight the differences between the two algorithms when implemented on FPGA. Both converters are dedicated for the special moduli set (2^n – 1, 2^n, 2^n + 1). This study has been published in the Electronics journal [21].

The new CRT-I is a parallel algorithm, but it requires a special modular adder (a rather large one), in order to compute the equivalent binary number. As shown in Fig. 3.9 (a), the reverse converter based on the new CRT-I requires two (2*n*-bit) CSA-EACs and a modulo $(2^{2n} - 1)$ adder. Utilizing CSAs results in a better timing performance of the design.

On the other hand, the reverse converter based on the MRC does not require any special large modular adders. Its structure is illustrated in Fig. 3.9 (b).

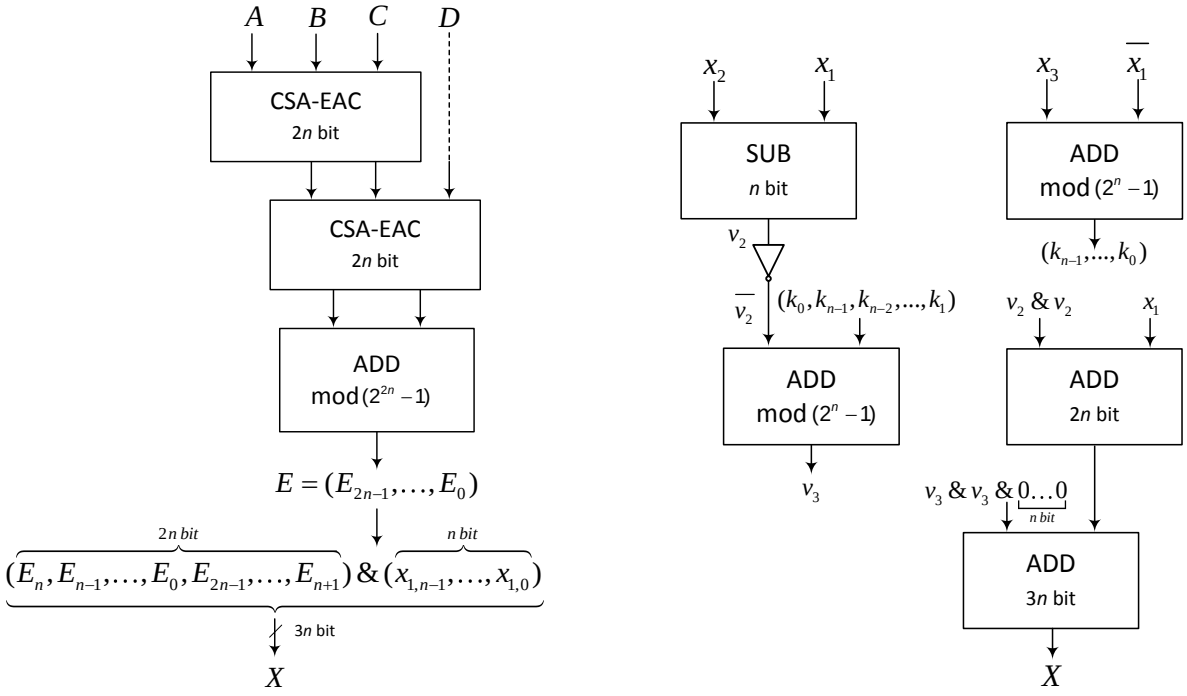


Fig. 3.9: Proposed structures of reverse converters [21], based on (a) the new CRT-I, (b) the MRC

The two converters were implemented on Spartan-3 xc3s200 FPGA. The implementation results have proven the theoretical considerations, that the new CRT-I is a parallel algorithm, but has more area consumption due to the usage of two 2*n*-bit CSAs and a large modular adder (modulo $(2^{2n} - 1)$ adder). Although the MRC is a sequential algorithm, it does not require any special large modular adders, which results in less area consumption but longer delay.

The new CRT-I has shorter critical path delay than that of the MRC. The difference between the two algorithms gradually increases as the dynamic range increases. Time saving percentage of the new CRT-I over the MRC has a maximum value (46.8%) for the very large dynamic range (60 bits). On the other hand, the MRC has considerably less hardware complexity than that of the new CRT-I. The difference between hardware complexity of the

new CRT-I and MRC gradually increases as the dynamic range increases. This difference achieves its greatest value (57.5 %) for the very large dynamic range (60 bits).

Thus, the MRC is preferred over the new CRT-I for designs with balanced and minimum-area strategies. Whereas, new CRT-I should be used for designs that have critical timing requirements.

3.4.2 Proposed algorithm for residue to binary conversion

In this section, a novel algorithm for reverse conversion based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is presented. The majority of papers regarding reverse converters are principally based on one of the widespread algorithms; the MRC, the CRT or the new CRTs. A paper, which presents the proposed algorithm, a reverse converter and a residue comparator based on it, is still under review in IEICE Electronics Express journal.

Proposed algorithm for reverse conversion

The main advantage of this algorithm is that, it does not need any multiplicative inverses neither multiplication processes. These calculations always have been the main obstacles in the reverse conversion methods.

The key concept of the proposed algorithm is that, the numbers within the dynamic range $[0, M - 1]$ can be divided into $(2^{2n} - 1)$ groups. According to this algorithm, a binary number X can be easily computed using the group number G and the residue x_2 corresponding to modulo (2^n).

$$X = G \times 2^n + x_2 \tag{3.1}$$

Proposed residue to binary converter based on the proposed algorithm

The structure of the proposed converter is shown in Fig. 3.10. The step that computes the group number G is realized using a read only memory (ROM). The size of this ROM is $(2^{n+1} \times n)$ bits. It contains the values needed to compute the correct group number that residues x_1, x_2 and x_3 belong to.

The proposed reverse converter was implemented on Virtex-4 XC4VSX25 FPGA. The proposed design was implemented for different dynamic range requirements, 12 bits, 15 bits (medium dynamic range), 24 bits and 33 bits (large dynamic range), 45 bits and 48 bits (very large dynamic range). The proposed design could not be implemented for dynamic ranges greater than 48 bits, due to the BRAM limitation integrated in Virtex-4 XC4VSX25.

The proposed reverse converter has been compared with another one based on the new CRT-I [3]. The implementation results concerning timing performance are very impressive. The proposed design can operate at higher frequencies up to 78.5%. Concerning area

consumption, the savings are not very impressive. The total number of the 4-input look-up tables used in the proposed converter is less for the dynamic ranges up to 33 bits. However, this number considerably increases for the very large dynamic ranges. Furthermore, the proposed design uses a number of BRAMS of 18 Kb. Nevertheless, the speed gain makes the proposed converter very attractive for further enhancements and improvements.

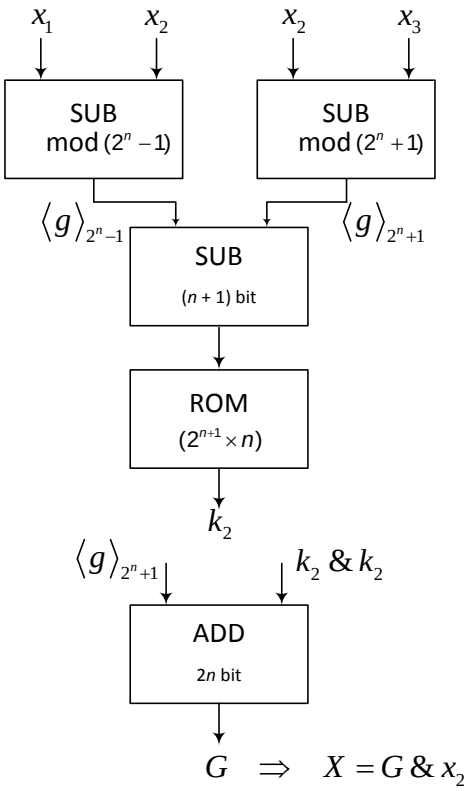


Fig. 3.10: The structure of the reverse converter based on the proposed algorithm [24]

The speed gain of the proposed reverse converter is about 23.4% for medium dynamic ranges. Then, it extensively increases to 78.5% for DR = 24 bits. This gain afterwards begins to gradually decrease until it reaches a break point (DR = 45 bits), where the speed gain becomes 0.2%. For DR = 48 bits, timing performance of the proposed reverse converter becomes worse than the new CRT-I based one [3]. However, according to my researches stated in [13] and [14], the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is not efficient to be used in applications that require very large dynamic ranges.

The proposed reverse converter has been also compared with pure ROM-based reverse converter. The size of this ROM-based reverse converter is $(2^{3n+1} \times 2n)$ bits. In a similar manner, this reverse converter has been implemented on Virtex-4 XC4VSX25 FPGA. This converter could only be implemented for DR = 12 bits and 15 bits, due to the limitations of BRAMs built-in this device. Therefore, it is obvious that the application fields of this type of

reverse converters are very limited. For DR = 12 bits, the maximum frequency is 401.3 MHz. It is obvious that this converter is abundantly faster than the proposed one. However, for DR = 15 bits, the maximum frequency of this converter considerably decreases (it turns out to be faster by 52.5%).

Hence, the above discussion leads to the fact that the proposed reverse converter is more efficient than both pure ROM-based and pure combinatorial ones. The main drawback of pure ROM-based converters is their size, whereas pure combinatorial converters suffer from poor timing performance comparing to the proposed one. However, for very large dynamic ranges (larger than 48 bits), these converters are more efficient.

3.5 Proposed residue comparator

The proposed algorithm for reverse conversion has been also used to compare residue numbers in their RNS representation. According to equation (3.1), in order to compare two RNS numbers (x_1, x_2, x_3), (y_1, y_2, y_3), the following values should be compared, (G_x and G_y) and (x_2 and y_2). This comparison is done using binary comparators of 2n bits and n bits. However, since G is the sum of k_2&k_2 and $\langle g \rangle_{2^n+1}$, the 2n-bit binary comparator is split into two parallel binary comparators of n bits and (n + 1) bits. The proposed structure of the residue comparator based on the proposed algorithm is shown in Fig. 3.11.

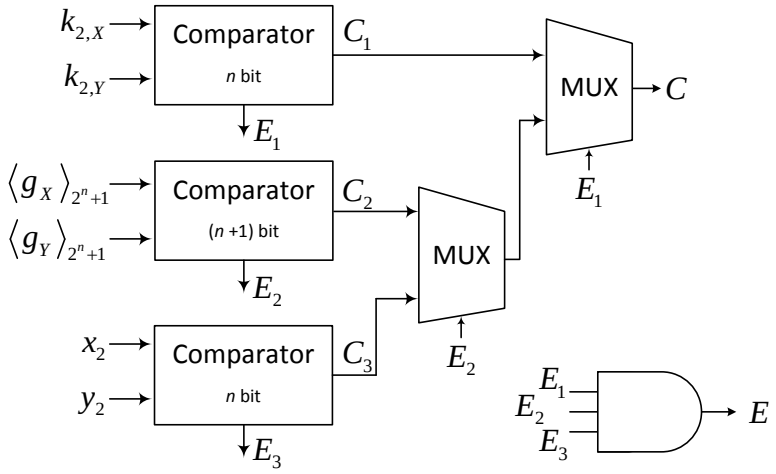


Fig. 3.11: The structure of the residue comparator based on the proposed algorithm [24]

The proposed residue comparator was implemented on Virtex-4 XC4VSX25 FPGA for different dynamic range requirements, 12 bits, 24 bits, 33 bits, 39 bits and 48 bits.

The structure of the proposed comparator is very similar to the one of the proposed reverse converter shown in Fig. 3.10. The only differences are the ROMs and the final 2n-bit adder. The proposed residue comparator has been compared with its counterparts [3], [6] for

different dynamic range requirements. The speed gain of the proposed comparator is about 9% for DR = 12 bits. Then, it extensively increases to 42.4% for DR = 33 bits. This gain afterwards begins to gradually decrease until it reaches a break point (DR = 39 bits), where the speed gain becomes 0.2%. From DR = 42 bits and larger, timing performance of the proposed reverse converter becomes worse than that of [6].

Hence, the implementation results show that the proposed comparator is not suitable to be used in RNSs that provide very large dynamic ranges. Again, we can overlook this fact since the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ is not efficient to be used in applications that require very large dynamic ranges.

3.6 Proposed designs for overflow and sign detection and correction in both signed and unsigned RNS systems

This section presents two universal efficient approaches for overflow and sign detection and correction in the addition of two numbers in unsigned and signed RNS. Both methods are designed to be used in systems based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ that provides an even dynamic range. Moreover, by applying a tiny modification, these designs can be used with any moduli set. The proposed designs are published in international conferences in Brno [19] and Seville, Spain [20].

3.6.1 Proposed design for overflow detection and correction in unsigned RNS systems

The general way to detect overflow is via comparing the sum with one of the addends, i.e. If $X \geq 0$ and $Y < M$ then $(X + Y) \bmod M$ causes overflow if and only if the sum is less than X. The proposed method also depends on comparison; however, it compares each of the addends with half of the RNS dynamic range ($M/2$).

To detect overflow during the addition of two addends X and Y in unsigned RNS based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, a single bit (*evlt_bit*), that indicates to which half of the dynamic range M that addend belongs, is used.

Fig. 3.12 illustrates the structure of the proposed design that detects the overflow in unsigned RNS based on $\{2^n - 1, 2^n, 2^n + 1\}$. The first component of this design is the magnitude evaluation unit, which evaluates the amplitude of the addends and their sum. It is realized by an *AND* gate with $2n$ inputs and an *OR* gate with two inputs. The second component is the overflow detection unit, which is realized by a 2:1 multiplexer and a *XOR* gate. The last component of the proposed design is the overflow correction unit. This unit adds back M to the sum (Z) in order to correct the overflow and obtain the final accurate result. The adder that performs the final addition can be of any type, according to the design's goal and strategy.

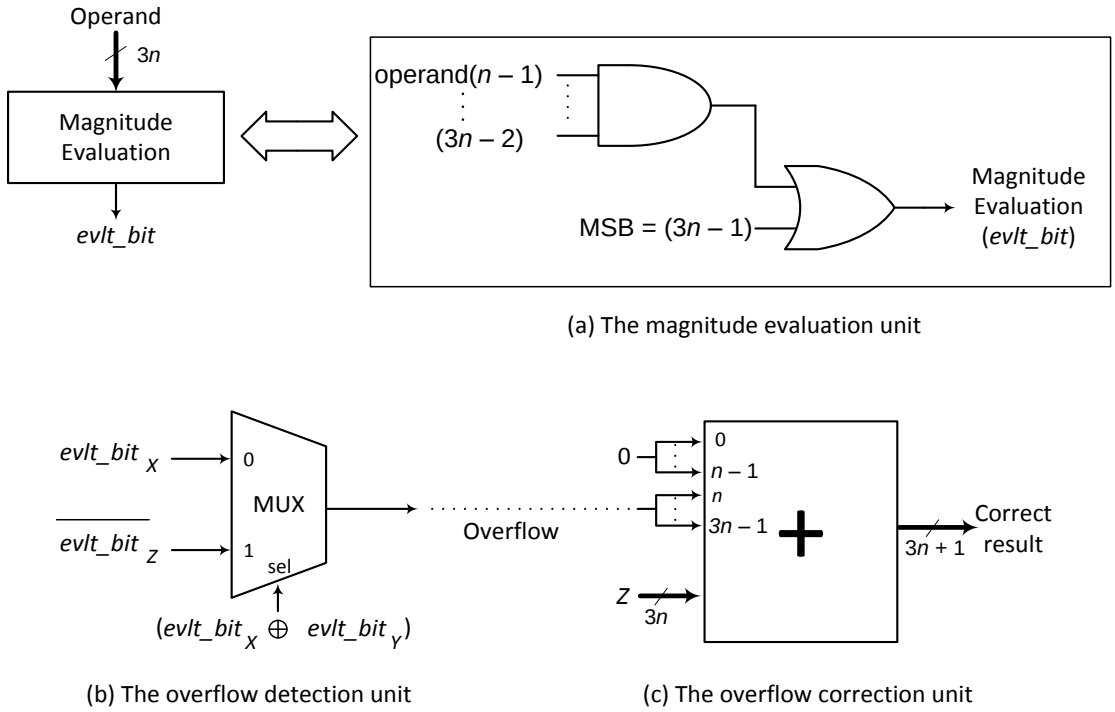


Fig. 3.12: The internal structure of the proposed overflow detection & correction unit for unsigned numbers [20]
(a) Magnitude evaluation unit. (b) Overflow detection unit. (c) Overflow correction unit

3.6.2 Proposed design for overflow and sign detection and correction in signed RNS systems

In a similar manner, to detect overflow and sign in the addition of two addends X and Y in signed RNS based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, a single bit (*evlt_bit*), that indicates to which half of the dynamic range M that addend belongs, is used.

In signed RNS, the positive numbers fall in the first half of the dynamic range, whereas, the negative ones fall in the second half. Thus, the sign evaluation of the addends is also represented by *evlt_bit*.

Fig. 3.13 shows the structure of the proposed design that detects the sign and overflow in signed RNS based on $\{2^n - 1, 2^n, 2^n + 1\}$. The first component of the proposed design is the sign evaluation unit, which evaluates the sign of the addends and their sum. It is realized by an identical structure to that of the magnitude evaluation unit of the proposed design for unsigned RNS. The second component is the overflow detection unit, which consists of a 2:1 multiplexer and two *XOR* gates. Finally, the last component is the overflow correction unit, which has an identical structure to that of the unsigned RNS. Similarly, the adder that performs the final addition can be of any type.

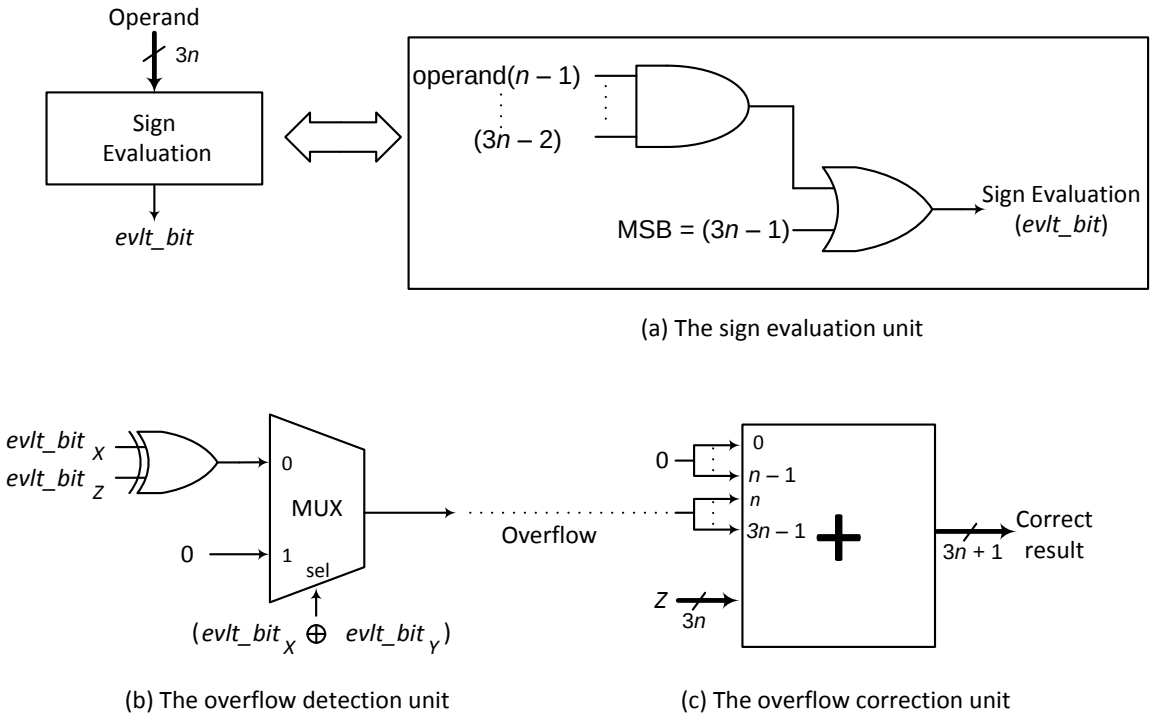


Fig. 3.13: The internal structure of the sign and overflow detection & correction unit for signed numbers [20] (a) Sign evaluation unit. (b) Overflow detection unit. (c) Overflow correction unit

3.6.3 Evaluating the proposed overflow and sign detection and correction designs

The proposed designs were compared with other two. The first one represents the general approach for overflow detection, which consists of a binary comparator based on the residue-to-binary converter presented in [3]. Whereas, the second one is an efficient residue comparator for general moduli set introduced in [6]. Both proposed designs have less delay and complexity without compromising on accuracy. Generally, this lower area requirements leads to lower power consumption. Moreover, by applying a tiny modification on the evaluation unit, both proposed designs can be used with any RNS that uses any moduli set.

3.7 Proposed RNS-based application

An RNS-based image processing application is presented in this section. This application applies a number of filters in spatial domain on a gray-scale image. This work has been published in Gdynia, Poland [22]. An extended version of this work is under review to be published in ElectroScope journal [23].

For performing filtering in spatial domain, a mask should be moved on the image. Theoretically, any spatial filter can be used based on the RNS, since the concept of its application is the same. During my research, I have implemented the following filters,

sharpen and edge detection. Using ROM-based converters turned out to be the most efficient method for this application and its dynamic range.

As shown in Fig. 3.14, the proposed design is divided into three stages; the first stage includes a ROM-based forward converter that converts the input pixels form binary to RNS. In the second stage, three parallel residue arithmetic units, corresponding to the three moduli, perform filtering operations (multiplying by the filter's coefficients and adding). The third stage converts the output residues of the three RAUs into their binary equivalent using ROM-based reverse converter.

According to [13], the most efficient moduli set for applications that require medium dynamic ranges (less than 22 bits) is $\{2^{n-1} - 1, 2^n - 1, 2^n\}$. We chose $n = 4$, so the used moduli set during our work is {7, 15, 16}. Its dynamic range = 1680 which is sufficient for image filtering application and eliminates the necessity to a special component for overflow detection and correction.

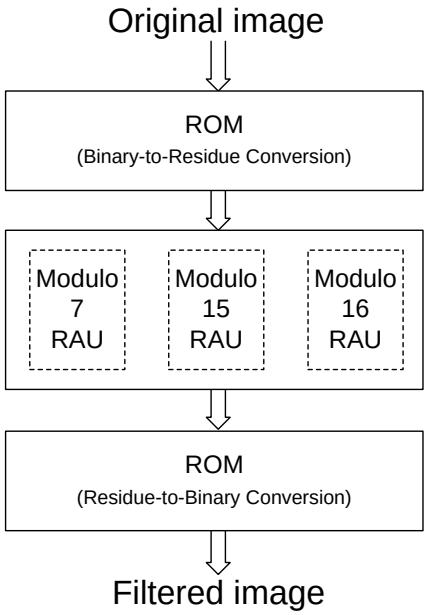
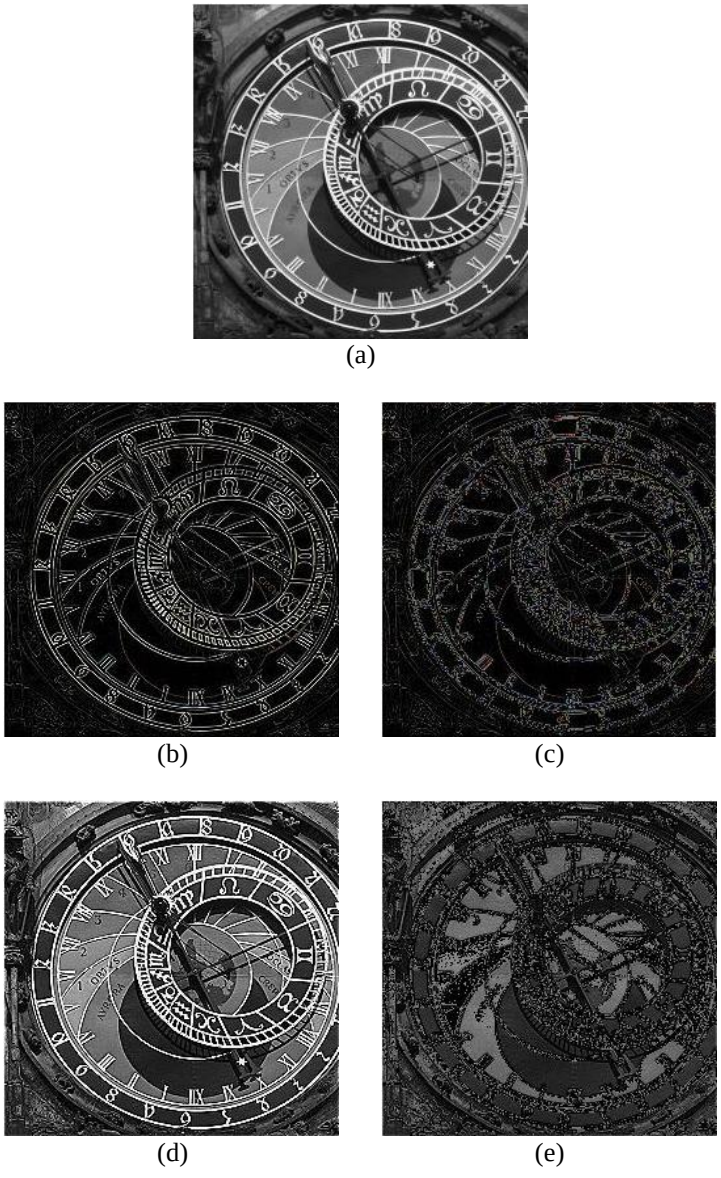


Fig. 3.14: The structure of the proposed RNS-based image-processing application [22]

The design was compiled and implemented on XC4VLX15 Virtex-4 FPGA device. A 256×256 gray-scale image was stored in a RAM. Both forward and reverse converters were implemented as ROMs. The proposed design was compared with its counterpart based on the binary number system. The proposed design has shown considerably more impressive characteristics than its counterpart. It can operate at higher frequency (by approx. 39.1%) and has less power consumption (by approx. 23.7%) when operating at frequency of 100 MHz.

On the other hand, the negative effects, of using an improper moduli set that provides an insufficient dynamic range for a digital image-processing application, are illustrated in Fig. 3.15. Many papers suggested using moduli sets with smaller dynamic ranges, such as {5, 7, 8} and {7, 8, 9} that provide M = 280 and M = 504, respectively [11], [10]. The authors of these papers considered that using these sets would be sufficient. However, the below figures clarify the fact that this is not true, except the case when using a special component for overflow detection and correction, which was not mentioned in any of those papers.



(a)

(b) (c)

(d) (e)

Fig. 3.15: The Output images after applying edge detection and sharpening filters,
(a) the original gray-scale image.
(b) after applying edge detection filter using the RNS based on the moduli set {7,15,16} [22].
(c) after applying edge detection filter using the RNS based on the moduli set {5,7,8} [11].
(d) after applying sharpening filter using the RNS based on the moduli set {7,15,16} [22].
(e) after applying sharpening filter using the RNS based on the moduli set {5,7,8} [11].

Fig. 3.15 shows the results of applying edge detection and sharpening filters using the RNS based on moduli sets {7, 15, 16} and {5, 7, 8}. The original input gray-scale image is shown in Fig. 3.15 (a). The output-filtered images using the RNS based on the moduli sets {7, 15, 16} and {5, 7, 8} are show in Fig. 3.15 (b) – (e). The negative effect of using a moduli set with insufficient dynamic range is clear. Fig. 3.15 (c) and (e) show the distorted output filtered images after applying edge detection and sharpening filters using the RNS based on the set {5, 7, 8}.

3.8 When to use the RNS (Binary vs. RNS)

This section can be considered as the conclusive outcome of this thesis. It illustrates the main issues that should be taken into account when deciding to use the RNS instead of binary number system.

During my research, I have observed that a number of published moduli sets do not provide better timing performance than that of binary-based application. Therefore, the first part of this section establishes the main aspect that should be considered when choosing a moduli set in order to achieve better timing performance than that of the binary number system. On the other hand, the second part studies the cases when utilizing the RNS would be the most advantageous. As aforementioned before, using the RNS is of great benefit in applications where addition, subtraction and multiplication are dominant. However, dominant is an abstracted word. Thus, a discussion about this issue is presented.

3.8.1 The effect of the critical modulo within a moduli set

The supreme feature of the RNS is speeding up arithmetic operations needed in many DSP applications. Many moduli sets of different forms and moduli number have been suggested. However, the unpredictable issue is that, using some of these moduli sets results in worse timing performance than that of binary-based ones.

During this research, I have concluded the main condition before choosing a moduli set for some applications. Briefly, in order to obtain better timing performance of addition or multiplication using the RNS, the critical modulo width (bits number) in the moduli set should be less than half of the dynamic range width this set provides. Hence, moduli sets that have such critical channels can be considered as meaningless, at least in applications that concern about timing performance. During my research, I have observed a number of already published papers suggesting moduli sets that can be disadvantageous to use $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ [5], $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ [7] and $\{2^{n/2} - 1, 2^{n/2} + 1, 2^n + 1, 2^{2n+1} - 1\}$ [8].

3.8.2 When is RNS superior than binary number system

As previously stated, the main advantageous field of using the RNS instead of binary number system is in applications that contain a dominant number of additions, subtractions and multiplications. Hence, this section discusses the issue of how many additions/multiplications should an application contain in order to obtain an RNS-based application faster than a binary-based one. Theoretically, in case of performing only one addition or multiplication, the binary-based application will be much faster than the RNS-based one, due to the delay of both forward and reverse converters. Therefore, in order to get the benefit of the RNS's properties, these operations should be performed at least a certain number of times in order to make the time-savings gained from performing these operations overweight the time consumed by the converters.

Two RNS-based applications, that perform additions and multiplications for different numbers of times, have been implemented on Virtex-4 XC4VSX25 FPGA. These applications are based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$.

The application based on iterated multiplications has shown much more impressive timing performance than the one based on iterated additions. Averagely, 10 iterated additions are required in order to obtain faster RNS-based application than binary-based one. On the other hand, only 2 iterated multiplications are enough to obtain faster RNS-based application than binary-based one.

Moreover, the speed gains in the case of iterated additions are not as impressive as those of iterated multiplications (e.g. the speed gain of RNS-based application over the binary-based one in case of 10 iterated additions for DR = 12 bits is 56.5%, whereas, it is 493% in the case of 10 iterated multiplications for the same DR).

Furthermore, the RNS-based application that performs iterated multiplications has shown tendencies to reduced-power consumption for dynamic ranges more than 33 bits. As the number of iterated multiplications increases, the percentage of power saving also increases (e.g. for DR = 48 bits, power saving percentage is 39.8% and 51.7% for 10 and 15 iterated multiplications, respectively).

Thus, it is evident that using the RNS is more advantageous in applications that have large and very large dynamic ranges and contain multiplications rather than only additions.

4 Conclusions

The main aim of this dissertation was designing RNS based building blocks for applications in the field of DSP applications (binary-to-residue converter, residue-to-binary converter, residue adder and residue multiplier). The achieved results and key outcomes are summarized in this Chapter.

Throughout this short thesis, a brief overview of the proposed work has been presented. The main RNS components have been introduced including a binary to residue converter, modular adders, modular subtractors, modular multipliers, a residue comparator, components for overflow and sign detection and correction and a residue to binary converter. The antithesis to the prevalent issue regarding the number of moduli within a set has been also presented. The three-moduli set $\{2^{n+1} - 1, 2^n, 2^n - 1\}$ [4] have shown the best timing performances among all other sets. Then, the sets that can be considered as pointless and the main condition for choosing a moduli set in order to obtain high timing performance are also discussed. Furthermore, a comparison between binary and RNS-based applications and the fundamentals that should be taken into account before designing a DSP application based on the RNS are also stated.

The efficiency of the proposed designs have been proven. The majority of the proposed components can be implemented with any system that has any moduli set of the form $(2^k \pm 1)$. Hence, the proposed building blocks can be implemented with any RNS system that uses any of the special moduli sets being published.

The proposed designs and outcomes of the doctoral thesis have been published in different national and international conferences and journals.

4.1 Final remarks

The final points of this thesis can be summarized as follows,

- The RNS-based applications should be used in fields that have dominant multiplications or at least a mix of multiplications and additions rather than only additions.
- The moduli set should be chosen in such a way that it contains as few moduli of the form $(2^k + 1)$ as possible, due to the complexity and delay caused by this channel. The most efficient set has been proven to be $\{2^{n+1} - 1, 2^n, 2^n - 1\}$.
- Contrary to the prevalent issue, the number of moduli within a set is not as important as it is widely known. Moduli number does not play a crucial role in enhancing the speed of the RNS system.

- Indeed, the form and magnitude of the largest modulo are the main concerns that should be taken into account. The width of the largest modulo should be at least less than half of the dynamic range‘s width.
- Enlarging the dynamic range is more efficient than using overflow detection units, since such components present a considerably long delay. Therefore, timing performance of the RNS-based application that uses these components is worse than the one that has a larger dynamic range and does not contain overflow detection units.
- Using the RNS in very large DRs results in so-called “super-efficient” applications compared to binary ones, (they provide considerably higher timing performance, reduced area and power consumption).
- Choosing the type of forward and reverse converters should depend on the dynamic range (more than 15 bits, the proposed converters have been proven to be more efficient than pure memory ones).

Hence, the points mentioned in the objectives of this thesis have been met. This work has led to new sights in the field of the residue number system. This system has very attractive and promising features if used in applications that require large and very large dynamic ranges. Rather than providing reduced power consumption with compromising timing performance or vice versa, both aspects can be fully met simultaneously. Since these two terms have become the key interests in nowadays technology, the RNS is the promising means that provides the so-called “super-efficient” applications.

Bibliography

[1] OMONDI, A., PREMKUMAR, B. *Residue Number System: Theory and Implementation*. London: Imperial College Press. 2007. 312 pages. ISBN-13: 978-1860948664.

[2] MOHAN, P.V.A., *Residue Number System: Algorithms and Architectures*. Massachusetts: Springer, 2002. 272 pages. ISBN-13: 978-1402070310.

[3] PIESTRAK, S.J. *A High-Speed Realization of a Residue to Binary Number System Converter*. In IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, 1995, vol. 42, p. 661 – 663. ISSN 1057-7130.

[4] MOHAN, P.V.A. *RNS-to-Binary Converter for a New Three-Moduli Set* $\{2^{n+1} - 1, 2^n, 2^n - 1\}$. In IEEE Trans. on Circuits and Systems-II: Express Briefs, 2007, vol. 54, p. 775 – 779. ISSN 1549-7747.

[5] MOLAHOSSEINI, A.S., NAVI, K., RAFSANJANI, M.K. *A New Residue to Binary Converter Based on Mixed-Radix Conversion*. In 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, p. 1 – 6. ISBN 978-1-4244-1751-3.

[6] BI, S., GROSS, W.J. *Efficient Residue Comparison Algorithm for General Moduli Sets*. In 48th Midwest Symposium on Circuits and Systems, 2005, vol. 2, p. 1601 – 1604. ISBN 0-7803-9197-7.

[7] WANG, W., SWAMY, M.N.S., AHMAD, M.O., WANG, Y. A Study of the Residue-to-Binary Converters for the Three-Moduli Sets. In IEEE Trans. on Circuits and Systems-I: Fundamental Theory and Applications, 2003, vol. 50, p. 235 – 243. ISSN 1057-7122.

[8] MOLAHOSSEINI, A.S., TEYMOURI, F., NAVI, K. *A New Four-Modulus RNS to Binary Converter*. In Proc. of IEEE International Symposium on Circuits and Systems, 2010, p. 4161 – 4164. ISBN 978-1-4244-5308-5.

[9] BEUCHAT, J.L. *Some Modular Adders and Multipliers for Field Programmable Gate Arrays*. In IPDPS '03 Proceedings of the 17th International Symposium on Parallel and Distributed Processing, 2003, ISSN 1530-2075.

[10] WANG, W., SWAMY, M.N.S., AHMAD, M.O. *RNS Application for Digital Image Processing*. In 4th IEEE international workshop on system-on-chip for real-time applications, 2004, vol. 1, p. 77–80. ISBN 0-7695-2182-7.

[11] TALESHMEKAEIL, D.K., MOUSAVI, A. *The Use of Residue Number System for Improving the Digital Image Processing*. In IEEE 10th International Conference on Signal Processing, 2010, vol. 1, p. 775–780. ISBN 978-1-4244-5897-4.

[12] ZIMMERMANN, R. *Efficient VLSI Implementation of Modulo* $(2^n \pm 1)$ *Addition and Multiplication*. In Proceedings of 14th IEEE Symposium on Computer Arithmetic, 1999, vol. 1, p. 158 – 167. ISBN 0-7695-0116-8.

Author's publications

[13] YOUNES, D., STEFFAN, P. *A Comparative Study on Different Moduli Sets in Residue Number System*. In International Conference on Computer Systems and Industrial Informatics, Dubai, UAE, 2012, vol. 1, p. 1 – 6. ISBN 978-1-4673-5155-3.

[14] YOUNES, D., STEFFAN, P., *A Detailed Study on the Moduli Number Effect on RNS Timing Performance*. In Journal of Emerging Trends in Computing and Information Sciences, Islamabad, Pakistan, 2013, vol. 4, p. 85 – 93. ISSN 2079-8407.

[15] YOUNES, D., STEFFAN, P. *Novel Architectures of Modulo* $2^n \pm 1$ *Adders for Field Programmable Gate Array*. In Electronic Devices and Systems IMAPS CS International conference, Brno, Czech republic, 2011, vol. 1, p. 51 – 56. ISBN 978-80-214-4303- 7.

[16] YOUNES, D., STEFFAN, P. *New Structures of* $2^n \pm 1$ *Modular Adders for FPGAs*. In ElectroScope Journal, Czech Republic, 2011, vol. 5, p. 11 – 14. ISSN 1313-1842.

[17] YOUNES, D., STEFFAN, P. *Improved Design for Modulo* $2^n + 1$ *Adder*. In Electronic Devices and Systems IMAPS CS International Conference, Brno, Czech republic, 2010, vol. 1, p. 346 – 348. ISBN 978-80-214-4138- 5.

[18] YOUNES, D., STEFFAN, P. *Novel Modulo* $2^n + 1$ *Subtractor and Multiplier*. In the Sixth International Conference on Systems ICONS, St. Maarten, the Netherlands Antilles, 2011, vol. 1, p. 36 – 38. ISBN 978-1-61208-002- 4.

[19] YOUNES, D., STEFFAN, P. *Efficient Method for Overflow Detection and Correction in Residue Number System*. In Electronic Devices and Systems IMAPS CS International Conference, Brno, Czech republic, 2012, vol. 1, p. 183 – 188. ISBN 978-80-214-4539- 0.

[20] YOUNES, D., STEFFAN, P. *Universal Approaches for Overflow and Sign Detection in Residue Number System Based on* $\{2^n - 1, 2^n, 2^n + 1\}$. In the Eighth International Conference on Systems, Seville, Spain, 2013, vol. 1, p. 1 – 5. ISBN 978-1-61208-246- 2.

[21] YOUNES, D., STEFFAN, P. *FPGA Implementation of Residue-to-Binary Converters: A Comparison between New CRT-I and MRC Converters for the Moduli Set* $(2^n - 1, 2^n, 2^n + 1)$. In Electronics Journal, Sofia, Bulgaria, 2011, vol. 5, p. 11 – 14. ISSN 1313- 1842.

[22] YOUNES, D., STEFFAN, P. *Efficient Image Processing Application Using Residue Number System*. In 20th International Conference Mixed Design of Integrated Circuits and Systems, Gdynia, Poland, 2013. p. 468 – 472. ISBN 978-83-63578-00- 8.

[23] YOUNES, D., STEFFAN, P. *Fast and Power Reduced RNS-Based Image Filtering in Spatial Domain*. In ElectroScope Journal, Czech Republic, 2013. ISSN 1313-1842. Under review.

[24] YOUNES, D., STEFFAN, P. *Efficient Reverse Converter and Residue Comparator Based on a Novel Algorithm in RNS*. In IEICE Electronics Express Journal. ISSN 1349-2543. Under review.

CURRICULUM VITAE

Name	Ing. Dina Younes
Date of Birth	30.05.1985
Contact	xyoune00@stud.feec.vutbr.cz
	+420775392607

Education and Training

2009 – present	Running for a Ph.D. degree, Department of Microelectronics, Faculty of Electrical Engineering and Communication, Brno University of Technology, Czech republic.
2009	Tempus exchange program, Department of Teleinformatics, Faculty of Electrical Engineering and Communication, Brno University of Technology, Czech republic.
2008 – 2009	Completed two semesters in Master of Teleinformatics, Faculty of Mechanical and Electrical Engineering, Tishreen University – Tempus Program, Latakia, Syria.
2003 – 2008	Engineer degree in Computer and Automatic Control Engineering, Faculty of Mechanical and Electrical Engineering, Tishreen University, Latakia, Syria.

Work Experience

2009 – present	Ph.D. student, junior researcher and Lab instructor (digital circuits and microprocessors in Czech and English) – Department of Microelectronics, Faculty of Electrical Engineering and Communication, Brno University of Technology, Czech republic.
2008 – 2009	Lab instructor (courses in digital image processing and C++ programming) – Department of Computer and Automatic Control Engineering, Faculty of Mechanical and Electrical Engineering, Tishreen University, Latakia, Syria.
Langauge Skills	Arabic, Russian, English, Czech, German

Abstract

This doctoral thesis deals with designing residue number system based building blocks to enhance the performance of digital signal processing applications.

The residue number system (RNS) is a non-weighted number system that provides carry-free, parallel, high speed, secure and fault tolerant arithmetic operations. These features make it very attractive to be used in high-performance and fault tolerant digital signal processing (DSP) applications.

A typical RNS system consists of three main components; the first one is the binary to residue converter that computes the RNS equivalent of the inputs represented in the binary number system. The second component in this system is parallel residue arithmetic units that perform arithmetic operations on the operands already represented in RNS. The last component is the residue to binary converter, which converts the outputs back into their binary representation.

The main aim of this thesis is to propose novel structures of the basic components of this system in order to be later used as fundamental units in DSP applications.

This thesis encloses improving and designing novel structures of these components, simulating and verifying their efficiency via FPGA implementation. In addition to suggesting novel structures of basic RNS components, a detailed study on different moduli sets that compares and determines the most efficient one for different dynamic range requirements is also presented. One of the main outcomes of this thesis is concluding and verifying the main condition that should be met when choosing a moduli set, in order to improve the timing performance of a DSP application. An RNS-based image processing application is also proposed. Its efficiency, in terms of timing performance and power consumption, is proved via comparing it with a binary-based one. Finally, the main considerations that should be taken into account when choosing to use the binary number system or RNS are also discussed in details.