Embedded Systems Design with Special Arithmetic and Number Systems

Author(s)	Molahosseini, Sabbagh Amir; de Sousa, Leonel Seabra; Chang, Chip-Hong
Imprint	Springer International Publishing, 2017
ISBN	9783319497426, 9783319497419
Permalink	https://books.scholarsportal.info/en/read?id=/ebooks/ebooks5/springer5/2019-08-14/4/9783319497426
Pages	3 to 17

Downloaded from Scholars Portal Books on 2020-10-11
Téléchargé de Scholars Portal Books sur 2020-10-11

Chapter 1
Introduction to Residue Number System: Structure and Teaching Methodology

Amir Sabbagh Molahosseini and Leonel Sousa

1.1 Introduction

The current embedded systems and wireless world applications require energy efficiency and fast computation [1, 2]. A lot of progress has been made on different aspects of electronic systems in this direction, from transistor to architecture level. However, arithmetic level operations still suffer from long carry-propagation chains. Although many computer arithmetic methods, such as parallel-prefix computations, have been introduced to reduce delay, the transition from traditional weighted two's complement number system to an efficient alternative number system can have a significant impact in computation systems performance. The residue number system (RNS) has been the most interesting and challengeable alternative number system in computer arithmetic for more than half a century [3–5]. Its main interesting feature is the ability of performing addition, subtraction, and multiplication without carry-propagation between residues [6, 7]. This exceptional ability makes the computer arithmetic researcher's target, i.e., carry-free arithmetic, achievable. However, there are some difficulties for efficiently implementing division, residue to binary conversion, sign detection, magnitude comparison, and overflow detection. Therefore, RNS has been mainly used in applications where dominant operations are addition, subtraction, and multiplication, such as digital signal processing [8] and

A.S. Molahosseini (✉)
Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, 7635131167 Iran
e-mail: amir@iauk.ac.ir

L. Sousa
Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento (INESC-ID), Instituto Superior Técnico (IST), Universidade de Lisboa, Lisbon 1000-029 Portugal
e-mail: las@inesc-id.pt

© Springer International Publishing AG 2017

A.S. Molahosseini et al. (eds.), *Embedded Systems Design with Special Arithmetic and Number Systems*, DOI 10.1007/978-3-319-49742-6_1

cryptography [9, 10]. Moreover, RNS has many other interesting properties, such as error detection/correction and fault-tolerant abilities [11], which makes it suitable for other type of applications, namely computer networks [12], DNA arithmetic [13], and cloud storage [14]. In recent years, many advances have occurred in both designing and applications targeting RNS [5].

In order to design an RNS-based digital system, first of all the dynamic range to cover has to be defined, i.e., the range of numbers that need to be handled. Then, an efficient moduli set should be selected according to that dynamic range and the type of targeted applications. A forward converter is required to interface RNS with the weighted representation of numbers. Then, independent modulo arithmetic channels perform computations on the residues in parallel. Finally, a reverse converter translates the resulting RNS number into a weighted representation, used by conventional digital systems. In addition to those essential RNS components, some optional components such as a sign detector, a magnitude comparator, and a scaler can be used in the RNS datapath according to the application's requirements [5]. In the rest of this chapter, different methods for designing all RNS parts and components will be introduced and recent work and advances discussed. Moreover, for the first time, a step-by-step RNS teaching methodology is introduced, which integrates hardware structures, implementations, and applications together with mathematical background for RNS. This teaching methodology can facilitate electrical and computer engineering researchers and students to more easily understand RNS concepts and designs, and motivate them to research new RNS architectures and apply it in further practical applications.

1.2 RNS Structure

The first step for designing an RNS system is to select an appropriate moduli set, given the dynamic range (the product of all moduli) and the degree of required parallelism (corresponding to the number of moduli in the set). The moduli set includes pair-wise relatively prime numbers. In general, RNS moduli sets can be organized into two main groups: *arithmetic-friendly* and *conversion-friendly* moduli sets. Conversion-friendly moduli sets result in simpler reverse converter structures, in contrast to arithmetic-friendly sets. If, for the target application, the rate of internal arithmetic operations to conversions is high, arithmetic-friendly should be selected and vice versa. Some hints and useful points about the method of selection and sample moduli sets can be found in [15–19].

The 2^n and $2^n \pm 1$ moduli are the most well known and efficient since they allow the design of simple hardware arithmetic units. Based on these popular numbers, lots of special moduli sets have been introduced for RNS. The most known is the traditional moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. Although the majority of research on RNS has been devoted to this three moduli set, the limited parallelism provided prevents its usage in nowadays high-performance computation systems.

Thus, extended RNS moduli sets such as $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1, 2^{n+1} - 1\}$ have been proposed. Note that in addition to special RNS moduli sets, general RNS moduli sets exist where each modulo can be favorably selected by the designer; for example, for prime moduli sets, where all the moduli are prime, modulo multiplications can be transformed to additions based on an isomorphism [20]. Note that moduli sets with a larger amount of moduli are usually considering for cryptography applications, which operate on very large numbers [21].

A general RNS structure is depicted in Fig. 1.1. The forward converter receives weighted integer operands and produces residues. For example, let us consider the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ and $n = 4$. The dynamic range is 4080, and the

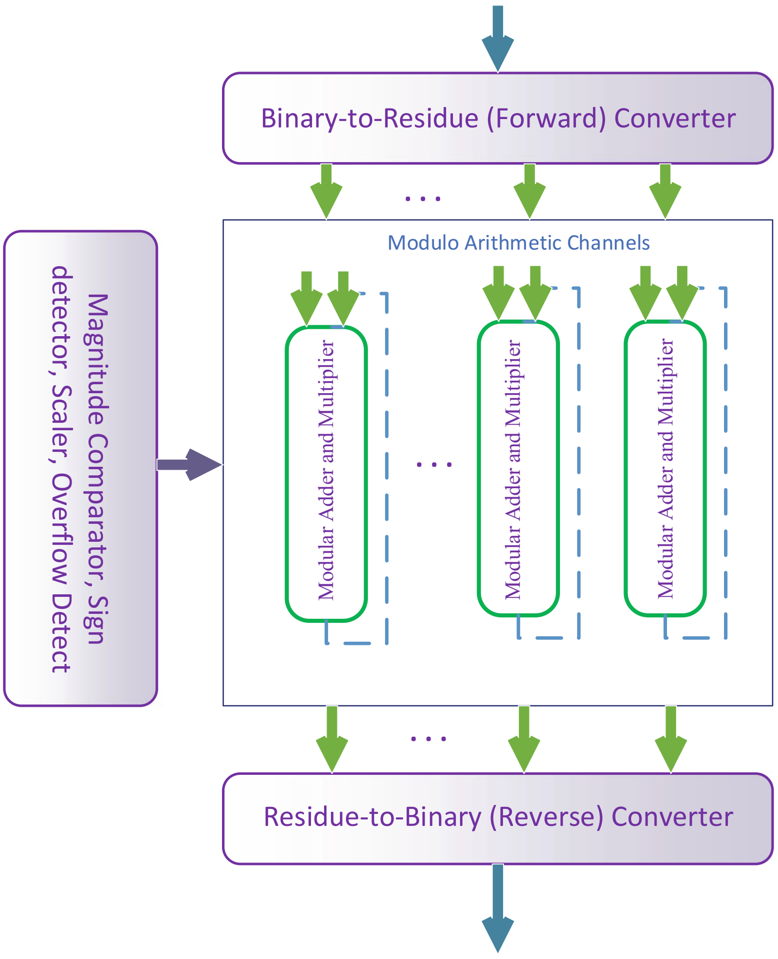
Binary-to-Residue (Forward) Converter
Modulo Arithmetic Channels
Modular Adder and Multiplier
Modular Adder and Multiplier
Modular Adder and Multiplier
Magnitude Comparator, Sign detector, Scaler, Overflow Detect
Residue-to-Binary (Reverse) Converter

Fig. 1.1 The RNS structure

residue representation of the initially weighted input $X = 20$ can be achieved by calculating the remainder of the division of X by the different moduli, resulting in $X = (5, 4, 3)$. In practice, one does not have to compute the entire process of division to obtain the residues, but only modular additions are required to obtain them for particular moduli sets [22]. Therefore, forward converters are parallel architectures, for the different residues, composed of independent units, each one computing one residue based on adders. The forward converter design has been addressed in many works, such as [23–31]. Moreover, recently, the design of forward converters for general RNS moduli sets has been considered in [32].

The design of a reverse converter design includes five phases. In the first phase, a conversion algorithm is selected according to the moduli set and its properties. In the subsequent phase, the values of the moduli together with the values of multiplicative inverses are considered in the conversion formulas. Then, residue arithmetic properties should be used to work on the conversion equations in such a way that makes the computation simpler. Next, the simplified equations are implemented using arithmetic circuits such as adders and logical gates. The most well-known conversion algorithms are: the Chinese remainder theorem (CRT), the mixed-radix conversion (MRC), and the New CRTs-I and CRT–II, derived from the original CRT. Details and examples about each of these algorithms can be found in [22]. There are many reverse converter designs for RNS moduli sets. Since the structure of the reverse converter is dependent on the moduli sets, special moduli sets with a limited number of moduli are usually used to design efficient reverse converters to optimize the design for different cost functions, such as performance, cost, and power consumption.

Modulo adder and multiplier designs are also an important research field in RNS because they are not only used in the parallel modulo arithmetic channels, but also in the forward and reverse converters. There are different architectures for designing adders and multipliers, for general and specific moduli [7], most of them for the moduli set $2^n - 1$, $2^n + 1$, and 2^n (for this last modulo regular binary adders and multipliers can be used [22]).

There are many other operations in RNS which may be required by the applications. One of these is sign detection. In contrast to the weighted number system, where the sign can be detected by the most significant bit (MSB), in RNS sign detection is hard, requiring the comparison of numbers with half of the dynamic range: integers in the first half of dynamic range are considered positive and all the other negative [7]. Furthermore, magnitude comparison is another hard operation in RNS. Usually, a semi-transformation of the RNS number to a weighted equivalent is applied to design RNS sign detectors and magnitude comparators. Other difficult operations are scaling and division. These are important operations, namely to avoid overflow in series of RNS multiplications and additions, which are typical operations of digital filtering. The scaling factor is usually considered as the power of two modulo to reduce the complexity of the operation. The overflow, that occurs when the result of an operation cannot be represented within the dynamic range, is also quite hard to detect in RNS. Table 1.1 introduces the most efficient and latest references that report hardware design for each RNS component.

Table 1.1 State-of-the-art RNS hardware components

Component	Moduli	Reference
Forward converter	$\{2^n-1, 2^n, 2^n+1\}$	[23–27]
	$2^n \pm 2^k + 1$	[28]
	$2^n \pm k$	[29]
	Generic	[30–32]
Adders and multipliers	$2^n \pm 1$	[33–41]
	Multi-modulus	[42–46]
	$2^n \pm k$	[47, 48]
	$2^n - 2^{\mathrm{k}} - 1$	[49]
	Generic	[50–52]
Reverse converter	$\{2^n-1, 2^n, 2^n+1\}$	[53]
	$\{2^n-1, 2^n, 2^{n+1}-1\}$	[54]
	$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$	[55–57]
	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	[58]
	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$	[59, 60]
	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$	[60, 61]
	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	[62]
	$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$	[63]
	$\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$	[64]
	Signed-output	[65]
Overflow detection	$\{2^n-1, 2^n, 2^n+1\}$	[66, 67]
Scaling	$\{2^n-1, 2^n, 2^n+1\}$	[68, 69]
	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	[70]
	$\{2^n-1, 2^{2n}, 2^n+1, 2^{2n+1}-1\}$	
Sign detection	$\{2^n-1, 2^n, 2^n+1\}$	[71]
	$\{2^n-1, 2^{n+k}, 2^n+1\}$	[72]
	$\{2^{n+1}-1, 2^n-1, 2^n\}$	[73]
	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$	[65]
	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	
Magnitude comparator	$\{2^n-1, 2^n, 2^n+1\}$	[74]
	$\{2^n-1, 2^{n+k}, 2^n+1\}$	[75]
	$\{2^{n+k}, 2^n-1, 2^n+1, 2^{n\pm1}-1\}$	[76]

1.3 RNS Teaching Methodology

For teaching RNS-based embedded systems, professors of embedded systems courses have to be aware of the latest improvements and enhancements at the arithmetic level. The concept of residue number-based embedded systems and the RNS potential are not widely known in this domain. Whereas the tutors who actively track publications may be aware of its potential, it is hard to integrate RNS in a systematic and organized way on embedded systems courses. The tutor should be aware of the features of each RNS component and its overall structure and abilities, as well as the method for applying it to embedded systems. Due to the multi-disciplinary nature of RNS, which is based on mathematical formulation, digital design, and computer architecture, just presenting a brief about the emerging approach of RNS-based embedded system design can confuse students and does not transmit them the ability of using modular arithmetic in RNS-based embedded systems. Moreover, the usage of RNS is manifold, and covering all its applications might not be possible. RNS can be used in both parts of embedded systems: hardware and software. The RNS structures can be embedded in single-purpose and application-specific processor architecture supported on extensions to the instruction-set architecture (ISA) [77]. Also, RNS is effectively used in parallel-programming of embedded systems as shown in RNS-based cryptography implemented on GPU [78].

With the aim of teaching the RNS approach for designing embedded systems and motivating students to work and do research on it, a systematic and organized methodology is introduced in this section. This methodology consists of two main phases: first, students should become familiar with the concept and structure of RNS. It is quite important to make students familiar with an exact and best fitted structure for embedded system applications. If only lots of references are introduced to the students without introducing an example of a selected architecture, the topic might be made arid. However, it is time consuming for a teacher to visit the complete set of RNS papers to find the best architecture that fits embedded systems and the course. Therefore, this section eases this process and introduces a simplified structure for students to learn each part in order to have a deep understanding of how to design the core of the arithmetic path of embedded processors based on RNS. Moreover, we also suggest some hints and introduce some references for researchers in the area of embedded systems who want to do more investigation in this area and start research projects.

We used the moduli set of $\{2^n - 1,\ 2^k,\ 2^n + 1\}$ as the base set to learn RNS for students, because the circuits for this RNS set are simple and there are also papers proposing embedded processor based on it [57, 77]. Although these conference papers can be pointed out as good examples of embedded systems based on RNS, there are differences in comparison with the architecture we follow here. Because students may not know well carry-save arithmetic, since they are embedded systems students and it is not a course on computer arithmetic. The second part of the proposed methodology is to learn the embedded processor

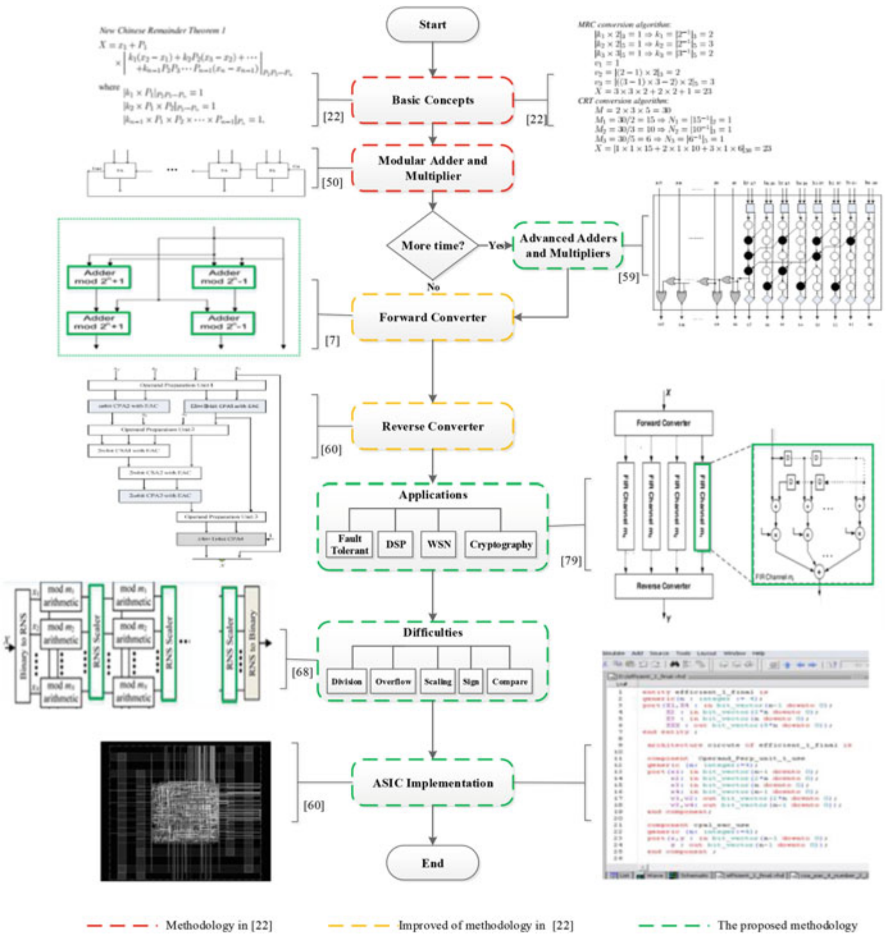

Fig. 1.2 The (hardware-oriented) methodology to teach RNS

architecture based on RNS, discuss the results, and show the superiority of RNS in real case studies through experimental results. In other words, students are taught that their knowledge in the embedded system course can be combined with modular arithmetic circuits to boost performance with high energy efficiency. The proposed methodology is depicted in Fig. 1.2, and its distinct parts are explained in the following subsections.

1.3.1 Basic Concepts

The first stage is devoted to reviewing the fundamental congruence theorems and formulas. The second chapter of the book [7] can be used with this aim, since it provides the most important formulas in a convenient way. It should be mentioned that numerical examples can play a significant role to make it easier for the students to understand the subject. Numerical examples should be used for all RNS parts, so that before achieving hardware implementations students understand the operation and functionality of each RNS part.

It is recommended that teachers first select a sample moduli set, and then show the residues of numbers in RNS in a table. Then, after each operation, the table can be used to show the correctness of operation. This is especially important in learning the difficult RNS operation's concepts. After introducing the dynamic range table, the tutor can select two RNS numbers without showing the table, and ask the students to compare them and recognize their signs. These easy questions make students familiar with RNS hard operations. Moreover, the lack of a sign bit, significance, and relation between residues can be mentioned to illustrate the difficulties in implementing inter-modulo operations. Numerical examples can be used as well as materials presented in [22] to facilitate student learning. At the end of this stage students should be able to understand the mathematical basis of RNS and the operation involved in each part, the moduli set selection procedure, and how to achieve simplified formulas of forward and reverse conversions for carefully selected moduli sets. It is also recommended that teachers give a brief explanation about RNS floating-point numbers [80]. If this point is well developed, it can be greatly suitable for energy-efficient floating-point units. This part of the curriculum was previously denoted by researchers [22]. Here, it is complemented by adding some important features such as a discussion about difficult RNS operations concepts, using a dynamic range table, and floating-point operations. Spending time only explaining formulas without considering details of hardware implementations may be the main reason for students to misunderstand the RNS capabilities, which are considered in more detail in the next sections.

1.3.2 Modular Adders and Multipliers

The second stage of the methodology is about modular adders and multipliers suitable for RNS. It is suggested that one first describes adders and then multipliers. Here, moduli $2^n - 1$ and $2^n + 1$ adders are presented and their operations are explained in detail, since they are significant and applicable in real implementations of RNS. The hardware architectures used to teach these adders can be ripple-carry adder (RCA)-based, because of their simplicity especially for beginners. Papers [33] and [50] can be good references in this stage. It is better to start with other moduli that have recently attracted research, such as $2^n \pm 3$, which can remain as a topic for the course's projects. If more time can be spent in this stage, more advanced structures such as hybrid adders and high-radix booth recoded modulo multipliers can be presented. Besides this, talking about the circuit's parameters of distinct structures and the impact of using different structures should be certainly considered also at this stage. Tutors should help students understand the advantages and disadvantages of the different structures, and prepare them for application-specific component selection. As an example, a teacher can introduce parallel-prefix modulo adders as high-speed adders but with large power consumptions. On the other hand, ripple-carry adders with end-around carry consume much less energy at the cost of higher delay. Explaining the hybrid structures of [59] and using the experimental results to show the trade-off between delay and power consumption can clearly show the impact of adder structures on circuits design and efficiency. Similar examples for multipliers can be picked up from [36]. In this paper, the authors increase the delay of the modulo $2^n - 1$ multiplier with the aim of reducing the area and power consumption, whenever the multiplier is not on the critical path. It is interesting for students to see how circuits' parameters may change the architecture of the multiplier.

1.3.3 Forward Converter

At the time students are familiar with adders and multipliers, one can explain the forward converter implementation. Since the forward converter is typically composed of multi-operand modular adders, it is suggested to teach this part after explaining adders, on the contrary to what was previously proposed [22]. Since understanding forward converters hardware is simpler after knowing well modular additions, the professors can easily show the complete architectures of forward converters for different moduli.

1.3.4 Reverse Converter

Afterwards, it's the moment to explain one of the most vital and complex operations in RNS. It is absolutely necessary students become familiar with the design and implementation of reverse converters. Reverse converters heavily depend on the moduli set. Many other factors such as the adopted algorithm and components also affect reverse converter performance. During the first stage, students became familiar with moduli set selection and different reverse conversion mathematical algorithms. Herein, the aim is to show the students how to implement simplified final reverse conversion equations using modular adders. It is not possible to teach this part in a general way, with its all details, because of the dependency between the converter architecture and the selected moduli set. Therefore, papers with specific moduli sets (see Table 1.1) can be a good choice for teaching this section. A helpful exercise for this part is to pick up another specific moduli set, and ask students to find the simplified formulas and implement them.

1.3.5 Application: RNS-Based Embedded Systems Design

After familiarizing the students with the interesting features and structures of the residue number system, it is possible to show them how RNS can be used in embedded system design. First, the teacher can depict a structure of a typical embedded system architecture with the usual parts, such as an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), a DSP, cryptography processing cores, and memory. Then, he can describe how to use the corresponding RNS structures to enhance the embedded system.

In the past, analog signals were converted to digital ones using regular ADCs, and then the digital signal transformed to the residues using forward converters. However, recently analog to residue converters (ARC) [81] have been designed in order to directly convert analog signals to digital residues. Therefore, the delay and cost of forward converters can be included in the ADC with this approach. However, all the elements of the system should operate over residues. Similarly, a digital residues to analog signal converter has been proposed (RAC) [82] that eliminates the need of a reverse converter. Various RNS-based DSPs [5] and cryptographic [10] processing circuits together with RNS-based memory [83] are available which can be used in the structure of an embedded system to achieve distinct applications.

At this point, the required concepts for understanding RNS as a part of a computer arithmetic and embedded system course are described. Additionally, if more time can be spent on teaching RNS, the two following parts can also be introduced into the proposed methodology.

1.3.6 Hard RNS Operations

In some computation-intensive applications, other operations such as sign detection, magnitude comparison, overflow detection, scaling, and division are required. These hard RNS operations were not covered in previous syllabus and books. Although there are suggested methods to overcome the difficulties and implement them, they have not yet been efficiently solved. Consequently, on-going research can be found in these fields. The tutors are suggested to explain the procedure of designing each of them based on recent papers (see Table 1.1). The advantage of explaining difficult RNS operations is not only to answer students' questions about them but also to motivate them to research on these challenging parts.

1.3.7 ASIC/FPGA Implementation

The final and optional, but also recommended, stage of the proposed methodology is showing the complete process of designing a hardware implementation of an RNS system, which includes the design, hardware description language (HDL) coding, functional verification using simulation tools, synthesis of the logical circuits, and experimental evaluation. Illustrating these phases using an example can provide a practical view to students and motivate them. It is supposed that students have background in digital circuits design and HDL coding. However, some brief descriptions about different types of HDL coding and the selection of either structural or behavioral modeling according to the situation can be helpful. Moreover, this stage can be a good opportunity for discussion about the effect of careful and wisely placement of different hardware components in the structure of RNS circuits to achieve suitable trade-offs between area, delay, and power consumption on application-specific integrated circuits (ASIC) or field-programmable gate arrays (FPGA). Note that it is not mandatory to implement all parts of RNS, since it is time consuming. It is enough to show a sample, e.g., prepared HDL codes for one important part of RNS, such as for the reverse converter, which can be complemented with ASIC or FPGA implementation results.

1.4 Conclusion

The RNS special abilities, such as carry-free arithmetic and fault-tolerant computations, fit high-performance modern embedded systems requirements. Research on RNS has been applied to different fields over the past decades, proving its importance. Unfortunately, the traditional RNS teaching methods focused on the mathematical foundations, which is not appealing for students interested in embedded systems. In order to solve this issue, a methodology is proposed, which integrates hardware structures, implementations, and applications together with the mathematical basis of RNS. By applying this method, it is easier for engineers to understand the potential of RNS, and motivate them to apply RNS for designing more efficient embedded systems.

References

1. P. Marwedel, *Embedded system design: embedded systems foundations of cyber-physical systems* (Springer International Publishing, Dordrecht, 2011)
2. S. Yang, Toward a wireless world. IEEE Technol. Soc. Mag. **26**(2), 32–42 (2007)
3. H.L. Garner, The residue number system. IRE Trans. Electron. Comput. **8**(2), 140–147 (1959)
4. T. Stouraitis, V. Paliouras, Considering the alternatives in low-power design. IEEE Circuits Devices **7**, 23–29 (2001)

5. C.H. Chang, A.S. Molahosseini, A.A.E. Zarandi, T.F. Tay, Residue number systems: a new paradigm to datapath optimization for low-power and high-performance digital signal processing applications. IEEE Circuits Syst. Mag. **15**(4), 26–44 (2015)
6. P.V.A. Mohan, *Residue number systems: Algorithms and architectures* (Kluwer, Boston, 2002)
7. A. Omondi, B. Premkumar, *Residue number systems: Theory and implementations* (Imperial College Press, London, 2007)
8. J. Chen, J. Hu, Energy-efficient digital signal processing via voltage-over scaling-based residue number system. IEEE Trans. Very Large Scale Integr. Syst. **21**(7), 1322–1332 (2013)
9. S. Antão, L. Sousa, The CRNS framework and its application to programmable and reconfigurable cryptography. ACM Trans. Archit. Code Optim. **9**(4), 1–33 (2013)
10. L. Sousa, S.F. Antão, P.S.A. Martins, Combining residue arithmetic to design efficient cryptographic circuits and systems. IEEE Circuits Syst. Mag., to appear, 2016
11. T.F. Tay, C.H. Chang, A non-iterative multiple residue digit error detection and correction algorithm in RRNS. IEEE Trans. Comput. **65**(2), 396–408 (2016)
12. R. Ye1, A. Boukerche, H. Wang, X. Zhou, B. Yan, RESIDENT: a reliable residue number system-based data transmission mechanism for wireless sensor networks, Springer Wireless Netw., to appear, 2016
13. X. Zheng, B. Wang, C. Zhou, X. Wei, Q. Zhang, Parallel DNA arithmetic operation with one error detection based on 3-moduli set, IEEE Trans. Nano Biosci., to appear, 2016
14. A. Celesti, M. Fazio, M. Villari, A. Puliafito, Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. J. Netw. Comput. Appl. **59**, 208–218 (2016)
15. M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, K. Navi, Efficient RNS implementation of elliptic curve point multiplication over GF(p). IEEE Trans. Very Large Scale Integr. Syst. **21**(8), 1545–1549 (2013)
16. Y. Liu, E.M.K. Lai, Moduli set selection and cost estimation for RNS-based FIR filter and filter bank design. Des. Autom. Embedded Syst. **9**, 123–139 (2004)
17. M. Dasygenis, I. Petrousov, A generic moduli selection algorithm for the residue number system, in *Proceedings of International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2015
18. A. Persson, L. Bengtsson, Forward and reverse converters and moduli set selection in signed-digit residue number systems. J. Signal Process. Syst. **56**(1), 1–15 (2009)
19. J.C. Bajard, M. Kaihara, T. Plantard, Selected RNS bases for modular multiplication, in *Proceedings of the 19th IEEE symposium on Computer Arithmetic*, 2009
20. J. Ramírez, U. Meyer-Bäse, F. Taylor, A. García, A. Lloris, Design and implementation of high-performance RNS wavelet processors using custom IC technologies. J. VLSI Signal Process. Syst. Signal Image Video Technol. **34**(3), 227–237 (2003)
21. F. Gandino, F. Lamberti, G. Paravati, J.C. Bajard, P. Montuschi, An algorithmic and architectural study on montgomery exponentiation in RNS. IEEE Trans. Comput. **61**(8), 1071–1083 (2012)
22. K. Navi, A.S. Molahosseini, M. Esmaeildoust, How to teach residue number system to computer scientists and engineers. IEEE Trans. Educ. **54**(1), 156–163 (2011)
23. F. Pourbigharaz, H.M. Yassine, Simple binary to residue transformation with respect to 2/sup m/+1 moduli. IEE Proc. Circuits Devices Syst. **141**(6), 522–526 (1994)
24. S. Piestrak, Design of residue generators and multioperand modular adders using carry-save adders. IEEE Trans. Comput. **43**(1), 68–77 (1994)
25. A.B. Premkumar, E.L. Ang, E.M.-K. Lai, Improved memoryless RNS forward converter based on the periodicity of residues. IEEE Trans. Circuits Syst. II Express Briefs **53**(2), 133–137 (2006)
26. C. Efstathiou, N. Moschopoulos, K. Tsoumanis, K. Pekmestzi, On the design of configurable modulo 2^n $\pm$ 1 residue generators, in *Proceedings of euromicro conference on digital system design (DSD)*, 2012, pp. 50–56
27. H.T. Vergos, D. Bakalis, C. Efstathiou, Fast modulo $2^n + 1$ multi-operand adders and residue generators. Integr. VLSI J. **43**(1), 42–48 (2010)

28. A.A. Hiasat, Arithmetic binary to residue encoders for moduli $(2^n \pm 2^k + 1)$. IEE Proc. Comput. Digit. Tech. **150**(6), 369–374 (2003)
29. P.M. Matutino, R. Chaves, L. Sousa, Arithmetic-based binary-to-RNS converter modulo $(2^n \pm k)$ for *jn*-bit dynamic range. IEEE Trans. Very Large Scale Integr. Syst. **23**(3), 603–607 (2015)
30. K. Shirakawa, T. Uemura, Y. Iguchi, A realization method of forward converters from multiple-precision binary numbers to residue numbers with arbitrary mutable modulus, in *Proceedings of IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, 2011
31. G. Petrousov, M. Dasygenis, A unique network EDA tool to create optimized ad hoc binary to residue number system converters, in *Proceedings of International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2014.
32. J.Y.S. Low, C.H. Chang, A new approach to the design of efficient residue generators for arbitrary moduli. IEEE Trans. Circuits and Syst.–I **60**(9), 2366–2374 (2013)
33. R. Zimmermann, Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication, in *Proceedings of the IEEE International Symposium on Computer Arithmetic*, 1999, pp. 158–167.
34. L. Kalampoukas et al., High-speed parallel-prefix modulo $2^n - 1$ adders. IEEE Trans. Comput. **49**(7), 673–680 (2000)
35. R.A. Patel, M. Benaissa, S. Boussakta, Fast parallel-prefix architectures for modulo $2^n - 1$ addition with a single representation of zero. IEEE Trans. Comput. **56**(11), 1484–1492 (2007)
36. R. Muralidharan, C.H. Chang, Radix-8 booth encoded modulo $2^n - 1$ multipliers with adaptive delay for high dynamic range residue number system. IEEE Trans. Circuits Syst.–I **58**(5), 982–993 (2011)
37. L.S. Didier, L. Jaulmes, Fast modulo $2^n - 1$ and $2^n + 1$ adder using carry-chain on FPGA, in Proc. of Asilomar Conference on Signals, Systems and Computers, 2013, pp. 1155–1159
38. R. Muralidharan, C.H. Chang, Area-power efficient modulo $2^n - 1$ and modulo $2^n + 1$ multipliers for $\{2^n - 1,\ 2^n,\ 2^n + 1\}$ based RNS. IEEE Trans. Circuits Syst.–I **59**(10), 2263–2274 (2012)
39. H.T. Vergos, G.N. Dimitrakopoulos, On modulo $2^n + 1$ adder design. IEEE Trans. Comput. **61**(2), 173–186 (2012)
40. H.T. Vergos, Area-time efficient end-around inverted carry adders. Integr. VLSI J. **45**(4), 388–394 (2012)
41. S.M. Mirhosseini, A.S. Molahosseini, M. Hosseinzadeh, L. Sousa, P. Martins, A reduced-bias approach with a lightweight hard-multiple generator to design radix-8 modulo $2^n + 1$ multiplier. IEEE Trans. Circuits Syst.-II, to appear, 2016
42. C.H. Chang, S. Menon, B. Cao, T. Srikanthan, A configurable dual moduli multi-operand modulo adder, in *Proceedings of IEEE International Symposium on Circuits and Systems*, 2005, pp. 1630–1633
43. E. Vassalos, D. Bakalis, H.T. Vergos, On the design of modulo $2^n \pm 1$ subtractors and adders/subtractors. Circuits Syst. Signal Process. **30**(6), 1445–1461 (2011)
44. C. Efstathiou, K. Tsoumanis, K. Pekmestzi, I. Voyiatzis, Modulo $2^n \pm 1$ fused add-multiply units, in *Proceedings of IEEE Computer Society Annual Symposium on VLSI,* 2015, pp. 91–96
45. H.T. Vergos, D. Bakalis, Area-time efficient multi-modulus adders and their applications. Microprocessors Microsystems Embedded Hardware Des. **36**(5), 409–419 (2012)
46. R. Muralidharan, C.H. Chang, Radix-4 and radix-8 Booth encoded multi-modulus multipliers. IEEE Trans. Circuits Syst.–I **60**(11), 2940–2952 (2013)
47. H. Pettenghi, S. Cotofana, L. Sousa, Efficient method for designing modulo $\{2^n \pm K\}$ multipliers. J. Circuits Syst. Comput. **23**(1), 1450001 (2014)
48. G. Jaberipur, S.H.F. Langroudi, (4 + 2\log n)\delta G parallel prefix modulo—$(2^n - 3)$ adder via double representation of residues in [0, 2]. IEEE Trans. Circuits Syst.-II **62**(6), 583–587 (2015)
49. S. Ma, J.H. Hu, C.H. Wang, A novel modulo $2^n - 2^k - 1$ adder for residue number system. IEEE Trans. Circuits Syst.-I **60**(11), 2962–2972 (2013)

50. J.L. Beuchat, Some modular adders and multipliers for field programmable gate arrays, in *Proceedings of Parallel and Distributed Processing Symposium*, 2003, pp. 1–8
51. R.A. Patel, M. Benaissa, N. Powell, S. Boussakta, Novel power-delay-area-efficient approach to generic modular addition. IEEE Trans. Circuits Syst.-I **54**, 1279–1292 (2007)
52. H. Nakahara, T. Sasao, A deep convolutional neural network based on nested residue number system, in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL)*, 2015, pp. 1–6
53. Y. Wang, X. Song, M. Aboulhamid, H. Shen, Adder based residue to binary numbers converters for $(2^n-1, 2^n, 2^n+1)$. IEEE Trans. Signal Process. **50**(7), 1772–1779 (2002)
54. 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. Circuits Syst.-II **54**(9), 775–779 (2007)
55. A. Hariri, K. Navi, R. Rastegar, A new high dynamic range moduli set with efficient reverse converter. J. Comput. Math. Appl. **55**(4), 660–668 (2008)
56. A.S. Molahosseini, K. Navi, O. Hashemipour, A. Jalali, An efficient architecture for designing reverse converters based on a general three-moduli set. J. Syst. Archit. **54**, 929–934 (2008)
57. R. Chaves, L. Sousa, Improving RNS multiplication with more balanced moduli sets and enhanced modular arithmetic structures. IET Comput. Digital Tech. **1**(5), 472–480 (September 2007)
58. P. Patronik, S.J. Piestrak, Design of reverse converters for general RNS moduli sets $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and $\{2^k, 2^n-1, 2^n+1, 2^{n-1}-1\}$ (n even). IEEE Trans. Circuits Syst.-I **61**(6), 1687–1700 (2014)
59. A.A.E. Zarandi, A.S. Molahosseini, M. Hosseinzadeh, S. Sorouri, S.F. Antão, L. Sousa, Reverse converter design via parallel-prefix adders: novel components, methodology and implementations. IEEE Trans. Very Large Scale Integr. Syst. **23**(2), 374–378 (2015)
60. A.S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, S. Timarchi, Efficient reverse converter designs for the new 4-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ based on new CRTs. IEEE Trans. Circuits Syst.-I **57**(4), 823–835 (2010)
61. L. Sousa, S. Antao, MRC-based RNS reverse converters for the four-moduli sets $\{2^n+1, 2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n+1, 2^n-1, 2^{2n}, 2^{2n+1}-1\}$. IEEE Trans. Circuits Syst. II **59**(4), 244–248 (2012)
62. H. Pettenghi, R. Chaves, L. Sousa, RNS reverse converters for moduli sets with dynamic ranges up to $(8n+1)$-bit. IEEE Trans. Circuits Syst. I **60**(6), 1487–1500 (2013)
63. C.C.H. Chang, T. Srikanthan, A residue-to-binary converter for a new 5-moduli set. IEEE Trans. Circuits Syst.–I **54**(5), 1041–1049 (2007)
64. M.H. Sheu, S.H. Lin, C. Chen, S.W. Yang, An efficient VLSI design for a residue to binary converter for general balance moduli $\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$. IEEE Trans. Circuits Syst.-II **51**(3), 152–155 (2004)
65. A.A.E. Zarandi, A.S. Molahosseini, L. Sousa, M. Hosseinzadeh, An efficient component for designing signed reverse converters for a class of RNS moduli sets with composite form {2K, 2P-1}, IEEE Trans. Very Large Scale Integr. Syst., to appear, 2016
66. L.C. Tai, C.F. Chen, Technical note. Overflow detection in a redundant residue number system. IEE Proc. Comput. Digital Tech. **131**(3), 97–98 (1984)
67. D. Younes, P. Steffan, Universal approaches for overflow and sign detection in residue number system based on $\{2^n–1, 2^n, 2^n+1\}$, in *Proceedings of Eighth International Conference on Systems*, 2013, pp. 77–81
68. C.H. Chang, J.Y.S. Low, Simple, fast and exact RNS scaler for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$. IEEE Trans. Circuits Syst.–I **58**(11), 2686–2697 (2011)
69. T.F. Tay, C.H. Chang, J.Y.S. Low, Efficient VLSI implementation of 2^n scaling of signed integer in RNS $\{2^n-1, 2^n, 2^n+1\}$. IEEE Trans. Very Large Scale Integr. Syst. **21**(10), 1936–1940 (2013)
70. L. Sousa, 2^n RNS scalers for extended 4-moduli sets. IEEE Trans. Comput. **64**(12), 3322–3334 (2015)
71. S. Kumar, C.H. Chang, A new fast and area-efficient adder-based sign detector for RNS $\{2^n-1, 2^n, 2^n+1\}$. IEEE Trans. Very Large Scale Integr. Syst. **24**(7), 2608–2612 (2016)

72. L. Sousa, P.S.A. Martins, Efficient sign identification engines for integers represented in the RNS extended 3-moduli set $\{2^n - 1, 2^{n+k}, 2^n + 1\}$. Electron. Lett. **50**(16), 1138–1139 (2014)
73. M. Xu, Z. Bian, R. Yao, Fast sign detection algorithm for the rns moduli set $\{2^{n+1} - 1, 2^n - 1, 2^n\}$. IEEE Trans. Very Large Scale Integr. Syst. **23**(2), 379–383 (2015)
74. Z. Torabi, G. Jaberipur, Low-power/cost rns comparison via partitioning the dynamic range. IEEE Trans. Very Large Scale Integr. Syst. **24**(5), 1849–1857 (2016)
75. L. Sousa, P.S.A. Martins, sign detection and number comparison on rns 3-moduli sets $\{2^n - 1, 2^{n+x}, 2^n + 1\}$, Circuits Syst. Signal Process., to appear, 2016
76. S. Kumar, C.H. Chang, T.F. Tay, New algorithm for signed integer comparison in $\{2\mathrm{n} + \mathrm{k}, 2\mathrm{n} - 1, 2\mathrm{n} + 1, 2\mathrm{n} \pm 1 - 1\}$ and its efficient hardware implementation, IEEE Trans. Circuits Syst.–I, to appear, 2016
77. R. Chokshi, K.S. Berezowski, A. Shrivastava, S.J. Piestrak, Exploiting residue number system for power-efficient digital signal processing in embedded processors, in *Proceedings of the International conference on Compilers, architecture, and synthesis for embedded systems*, 2009, pp. 19–28
78. S.F. Antão, J.C. Bajard, L. Sousa, RNS based elliptic curve point multiplication for massive parallel architectures. Comput. J. **55**(5), 629–647 (2012)
79. E. Vassalos, D. Bakalis, CSD-RNS-based single constant multipliers. J. Signal Process. Syst. **67**(3), 255–268 (2012)
80. J.S. Chiang, M. Lu, Floating-point numbers in residue number systems. Comput. Math. Appl. **22**(10), 127–140 (1991)
81. C.H. Vun, A.B. Premkumar, W. Zhang, A new RNS based DA approach for inner product computation. IEEE Trans. Circuits Syst.-I **60**(8), 2139–2152 (2013)
82. O. Abdelfattah, Data conversion in residue number system, M.E. Thesis, Department of Electrical and Computer Engineering, McGill University, Montreal, 2011
83. N.Z. Haron, S. Hamdioui, Redundant residue number system code for fault-tolerant hybrid memories. ACM J. Emerging Technol. Comput. Syst. **7**(1), 19 (2011)