A BINARY TO RESIDUE CONVERSION USING NEW PROPOSED NON-COPRIME MODULI SET

Mansour Bader[1], Andraws Swidan[2] , Mazin Al-hadidi[3] and Baha Rababah[4]

[1,2]Department of Computer Engineering, Jordan University, Amman, Jordan
[3]Department of Computer Engineering, Al-Balqa'a Applied University, Salt, Jordan
[4]Department of Computer Engineering, University of Portsmouth, Portsmouth, UK

ABSTRACT

Residue Number System is generally supposed to use co-prime moduli set. Non-coprime moduli sets are a field in RNS which is little studied. That's why this work was devoted to them. The resources that discuss non-coprime in RNS are very limited. For the previous reasons, this paper analyses the RNS conversion using suggested non-coprime moduli set.

This paper suggests a new non-coprime moduli set and investigates its performance. The suggested new moduli set has the general representation as $\{2^n-2, 2^n, 2^n+2\}$, where $n \in \{2,3,.....,\infty\}$. The calculations among the moduli are done with this n value. These moduli are 2 spaces apart on the numbers line from each other. This range helps in the algorithm's calculations as to be shown.

The proposed non-coprime moduli set is investigated. Conversion algorithm from Binary to Residue is developed. Correctness of the algorithm was obtained through simulation program. Conversion algorithm is implemented.

KEYWORDS

Forward Conversion, Residue Number System, Non-coprime Moduli Set

1. INTRODUCTION

Residue number system (RNS) is a subfield of finite field arithmetic [1]. It is widely used in digital signal processing, image processing, FIR (Finite Impulse Response) filters, and IIR (Infinite Impulse Response) filters because it is a carry-free system and high efficient in addition and multiplication [2]. So, residue number system is used by most applications that need a high degree of concurrency. A lot of researches in computer systems are enthusiastic to go through residue numbering system because of its characteristics such as, error detection and correction (fault tolerant) [3], modularity, and embedded parallelism.

RNS allows dividing a large number into smaller sub numbers. Numbers are represented by tipples which need less number of bits. The bits can be processed individually and in parallel without carry between them. This improves computation time and simplifies hardware implementation cost.

RNS has also the following advantages over conventional binary number system:

- Reducing the hardware complexity because the system is implemented by designing smaller processing units.

DOI : 10.5121/sipij.2016.7301

- Improving the speed of operations since all of the tasks are performed in parallel.
- Efficient realization of the various building blocks needed such as adders, multipliers.
- The absence of carry propagation between the modulus channels makes the RNS appealing for building parallel fast processors. This facilitates the realization of high-speed, low-power arithmetic. This advantage is of paramount importance in embedded processors, especially those found in portable devices, for which power consumption is the most critical aspect of the design [4].

Because of these features, computer arithmeticians have historically promoted the RNS for high-speed arithmetic-intensive applications [5].

The rest of this paper is organized as follows. In Section 2, overview of the new algorithm of forward conversion is proposed. Section 3 presents the new non-coprime realization of the proposed forward converter. The hardware implementation of the moduli set is presented in section 4, while the paper is concluded in Section 5.

2. NEW ALGORITHM OF THE FORWARD CONVERSION OVERVIEW

In working with RNS the following three main terminologies are used:

1. Moduli set: defined in terms of relatively prime moduli where the i^{th} modulus presented by m_i and the $\gcd(m_i, m_j) = 1, j \neq i, i = 1,2, \dots, n$. Numerous moduli sets can be used. The characteristics of RNS based systems depend on the moduli set chosen.

2. Dynamic range (M): this is equal to the product of m_i terms; $M = \prod_{i=1}^{n} m_i$, and denotes the interval of integers that can be represented uniquely in the RNS using the specific moduli set.

3. Residues: to represent any number X in RNS we find $x_i = X \bmod m_i$ for all m_i moduli. The number is represented as $X = x_1, x_2, x_3$.

RNS based processing units are generally composed of: Forward convertor, arithmetic and logic unit (ALU) and a reverse convertor shown in figure 1 [6].

Conversion from Binary to Residue is called forward conversion. This conversion is used in order to process numbers in Residue format, because it is faster and it is easier for human being understood. To use the Residue Number System efficiently one has to interface it with the real world, the numbers should be converted from usual representation either binary or analog to residue representation, this is the first step in using RNS. This step is a very complex and demanding process, which acts as an academic challenge that restrict the use of RNS in many practical applications. Many researches conducted to find the most efficient algorithm, hardware architectures and schemes either using special or arbitrary moduli sets in the implementation of forward convertors in RNS.

In the forward translation of a binary number to its RNS equivalent, one of the most trivial, classical expensive ways is to store all the residues and recall them based on the value of the binary input.

Using the fact that the number can be represented as:

$$X = x_{n-1}\, x_{n-2} \ \ldots x_1 x_0 = \sum_{j=0}^{n-1} x_j 2^j \qquad (1)$$

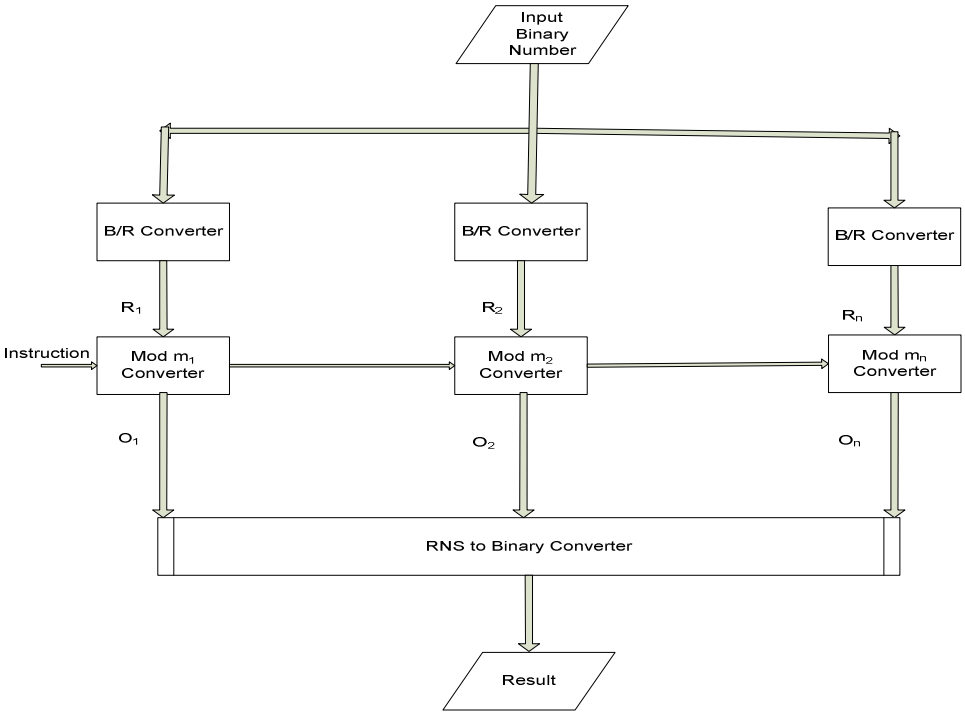


Figure 1. Block diagram of RNS based processing.

It is clear that this is a memory consuming process where you have to store all values in a lookup table that typically consists of ROM, and for complex applications that require large number representation, the size of memory will increase dramatically and thus increasing the cost.

The other implementation is to have special moduli set representation used in the conversion process, consist of three, four, or five bits, the design of these convertors is based on using carry save adder (CSA).

2.1. Non-coprime Moduli Sets Overview

Initially, only RNS with co-prime moduli set was investigated and used. Non-coprime had drawn the attention of research only lately. Thus the knowledge of non-coprime characteristics has been obtained from the co-prime one's as going to be seen through this chapter sections.

It is relatively easy to convert even numbers into their residue numbers representation. Numbers as 2^n, like 16 which is pow(2,4) or in form that we are familiar with now 2^4, and 14 which is $2^4 - 2$ and 18 which is $2^4 + 2$, are not co-primed with each other since there is a common factor between them which is number 2.

The usual way to compute **a mod m** is to take the remainder after integer division. This is straight forward when the operands are within the range of the available divide hardware, but the divide operation is known to be a slow arithmetic operation. Some small microcontrollers have no divide

hardware, and it is occasionally necessary to divide very large numbers outside the range that can be done using the available hardware [7].

It can be faster to take the modulus directly than to use the divide instruction when the modulus m is constant, even where there is a hardware divide instruction. Rules of divisibility mentioned in table 1 become even more valuable on machines without a hardware divide instruction or where the numbers involved are out of range.

Table 1. Some of divisibility check rules applied to decimal system.

Divisibility by n	**Check**
10	The least significant decimal digit is zero.
2	The least significant decimal digit is even.
5	The least significant decimal digit is 0 or 5
3	Sum of the decimal digits is divisible by 3.
9	Sum of the decimal digits is divisible by 9

2.2. Some Math Identities Review

2.2.1. Single General Rule

All divisibility check rules mentioned in table 1 are actually special cases of a single general rule.

Given that:

a is represented in number base b

a mod m = ((b mod m)(a/b) + (a mod b)) mod m

In the case of divisibility by 2, 5 and 10 for base 10, the term (b mod m) is zero because 2, 5 and 10 all divide evenly into 10. As a result, the divisibility test simplifies to asking whether (a mod b), that is, the least significant digit of the number, is evenly divisible.

In the case of divisibility by 3 or 9 in base 10, the term (b mod m) is one. As a result, the multiplier for the first term is one. Applying the formula recursively leads to the simple sum of the digits [7].

2.2.2. The Trivial Case: Mod 2, Mod 4, Mod 2^n

Computing modulus for powers of two is trivial on a binary computer, the term (b mod m) is zero, so we just take the modulus by examining the least significant i bits of the binary representation:

$a \bmod 2^i = a \,\&\, (2^i - 1)$

Thus, for a mod 2, we use a & 1, for a mod 4, we use a & 3, and for a mod 8, we use a & 7.

Recall that the **&** operator means logical and. When applied to integers, this computes each bit of the result as the and of the corresponding bits of the operands. For all nonzero positive integers i,

the binary representation of $2^i - 1$ consists of i consecutive one bits, so anding with $2^i - 1$ preserves the least significant i bits of the operand while forcing all more significant bits to zero [7].

The problem is more interesting when the modulus is not a power of two only.

2.2.3. Mersenne's Number: Mod 3, Mod 7, Mod 2^n-1

In mathematics, a Mersenne prime is a prime number that is one less than a power of two, i.e. 2^n-1. That is, it is a prime number that can be written in the form $Mn = 2^n - 1$ for some integer n.

They are named after Marin Mersenne, a French Minim friar, who studied them in the early 17th century [8].

Consider the problem of computing a mod 3 in binary number system. Note that 4 mod 3 is 1, so:
a mod 3 = ((a/4) + (a mod 4)) mod 3

That is, a mod 3 can be computed from the sum of the digits of the number in base 4. Base 4 is convenient because each base 4 digit of the number consists of 2 bits of the binary represenation; thus a mod 4 can be computed using a & 3 and a / 4 can be computed using a >> 2.

The number 3 is a Mersenne number, that is, one less than a power of two. The property noted above is true of all Mersenne numbers. Thus, we can compute a mod 7 or a mod 15 on a binary computer using:

a mod 7 = ((a/8) + (a mod 8)) mod 7

a mod 15 = ((a/16) + (a mod 16)) mod 15

Recall that a >> b shifts the binary representation of **a** left a total of **b** places. As with logical and, this is a very inexpensive operation on a binary computer, and the effect is the same as dividing a by 2b [7].

In this paper the problem is more interesting when the modulus is in a different shape of a power of two, where it is in Mod $(2^n - 2)$, Mod $(2^n + 2)$ consequently as a new moduli set proposed along with Mod (2^n), that are going to be discussed in the next section.

Our work is done by suggesting the new moduli set $\{2^n - 2, 2^n, 2^n + 2\}$ and proposing new conversion algorithms upon this new non-coprime moduli set. The next coming section will discuss the background of non-coprime moduli sets.

2.3. Our New Non-coprime Moduli Set Overview

As its name shows "non-coprime" means the non-coprimality among its modulus numbers. Non-coprime moduli sets can be used for error detection and correction purposes [9]. This non-coprimality could be shown in theorems and by examples to prove them too.

Observation 1: $2^k - 2$ is not relatively prime to 2^k, where k is a positive natural number.
It is obvious that 1 is not the only prime divisor of 2^k and $2^k - 2$, since they are both even and $2^k - 2$ is a multiple of 2 (i.e. a number multiplied by 2). Thus, there is a common divisor of them rather than 1, which is number 2 in this case. So $2^k - 2$ is not relatively prime to 2^k.

Example 1:

Let us take K = 4, in this case the two numbers of the theorem one would be 16 and 14 consequently, and it is obvious that there is a common divisor which is 2 between them when we try to bring them back to their elementary elements.

Observation 2: 2^k+2 is not relatively prime to 2^k, where k is a positive natural number.

It is obvious that 1 is not the only prime divisor of 2^k and 2^k+2 , since they are both even and $2^k + 2$ is a multiple of 2 (i.e. a number multiplied by 2). Thus, there is a common divisor of them other than 1, which is number 2 in this case. So 2^k+2 is not relatively prime to 2^k.

Example2:

Again let us take k = 4, in this case the two numbers of the theorem two would be 16 and 18 consequently, and it is obvious that there is a common divisor which is 2 between them when we try to bring them back to their elementary elements.

2.4. Properties of Non-coprime Moduli Set

2.4.1 Dynamic Range of Our Non-coprime Moduli Set

This property of having a common divisor other than number 1 led to the non-coprime moduli set when gathering the two theorems above. Based on these theorems the moduli set {2n - 2, 2n, 2n + 2} is non-coprime. The dynamic range of co-prime moduli set (M) is equal to $M = \prod_{i=1}^{n} m_i$. In the non-coprime case it is $M = (\prod_{i=1}^{n} m_i)/4$, it is 1/4 of the size of the co-primed one {2n – 1, 2n, 2n + 1}. This quarter comes from the multiplication of the common divisor (i.e. 2) between its modulus numbers leading to number 4 which can not be multiplied to form the usual (M), thus its size is less than the co-prime one. However it is important to know other characteristics such as its uniqueness and bits representation.

Definition [9]: We define a non-prime moduli set as (ml, ..., mk), where gcd (mi,mj) = l may not be satisfied for some i and j. The least common multiple of the moduli (m1, ..., mk), denoted as: M = lcm(m1, ..., mk), is the dynamic range of (m1, ..., mk) . For any decimal number $y \in [O, M – 1)$, y has a unique representation as (y1, ...,yk), where $y = yi \bmod mi$, $0 \leq yi < mi$.

If the residue number (y1, ...,yk) is consistent, the decimal number it represents can be found using the CRT theorem, where M = lcm(m1, ..., mk) [9].

To find M through a formula as the one of the co-prime, we suggest the following formula due to the special non-coprime moduli set:

$$M = (\prod_{i=1}^{n} m_i)/4 \qquad (2)$$

2.4.2 Uniqueness Verification

After dealing with the major and most important part of RNS in the previous section, it is important to discuss the moduli set uniqueness when converted from decimal form (as we see it) into RNS form (as to be implemented). In this section a mathematical proof is going to be presented powered by table 2 to show the uniqueness itself.

As a mathematical proof, we already know that 6, 8 and 10 are not co-primed with each other, in this case the LCM should be used as the previous section showed. Thus we have to return them back into their fundamental factors, for 6 which equal 2*3, for 8 it equals 23 and for 10 it is 2*5, now by taking the non common numbers without repetition and multiplying them we see that: 3*23*5 which equals 120, so M = 120. This is true for all other moduli sets chosen with the form (2n-2 ,2n, 2n +2), and this is the non-coprime moduli set that we have got and found our research on it for thesis level responsibilities, since the pre-moduli value (i.e. 2n -2) and the post moduli one (i.e. 2n +2) are multiples of 2 and are un-coprimed with each other, but they are co-prime with each other without it (as in 3 and 5 for the example above, when neglecting the multiplication of them with number 2). This fact is also true for all other moduli sets used; this is because they are 2 numbers relatively co-primed with each other and they are always odd.

The uniqueness interval was verified by simulation program, the results of a program using the moduli set (6,8,10) was simulated, at which each iteration is repeated every 6*8*10/4 times, which is in this case the number 120 (that is 'M' for the moduli set itself), and thus the uniqueness is guaranteed under the non-coprime moduli set within its dynamic range.

3. FORWARD CONVERSION ALGORITHM

This section has three sub sections of it as the number of the modulus numbers forming the moduli set are three.

3.1 Pre-modulus Algorithm Implementation

The binary representation led to this part's discovery, since its values are prime after dividing it by 2. We will give you how it works in words first then the algorithm of calculation $|X|_{2^n-2}$ can be represented in figure 1.

The idea of its binary cutting circles around the common factor which is number 2 in this case, as the previous sections showed that after dividing the pre number by the common factor the result is $2^{n-1} - 1$. Since the pre value has a $2^n - 2$ shape, then the cut of its binary representation would be in (**n** cut at the beginning as the co-prime algorithm did, but for the rest part it is taken (**n-1**) each time) starting from LSB again.

3.2 Middle-modulus Part Algorithm

No change is done on it, so we refer you to [15] – [17] where obtaining the residue of X with respect to modulus 2^n is the easiest operation.

3.3 Post-modulus Algorithm Implementation

The binary representation led to its working algorithm, since its value is prime after dividing it by 2. Again we will give you how it works in words first then we will put them in the flowchart of figure 2 to easily understand it.

The idea of its binary cutting circles around the common factor which is number 2 in this case, as the pre-modulus section showed. Here dividing the post number by the common factor the result is $2^n-1 + 1$, and since the post value has a $2^n + 2$ shape, then the cut of its binary representation would be in (n cut at the beginning as the co-prime algorithm did, but for the rest parts it is taken (n-1) each time) starting from LSB again as the pre-modulus did, some examples will show how this is done.

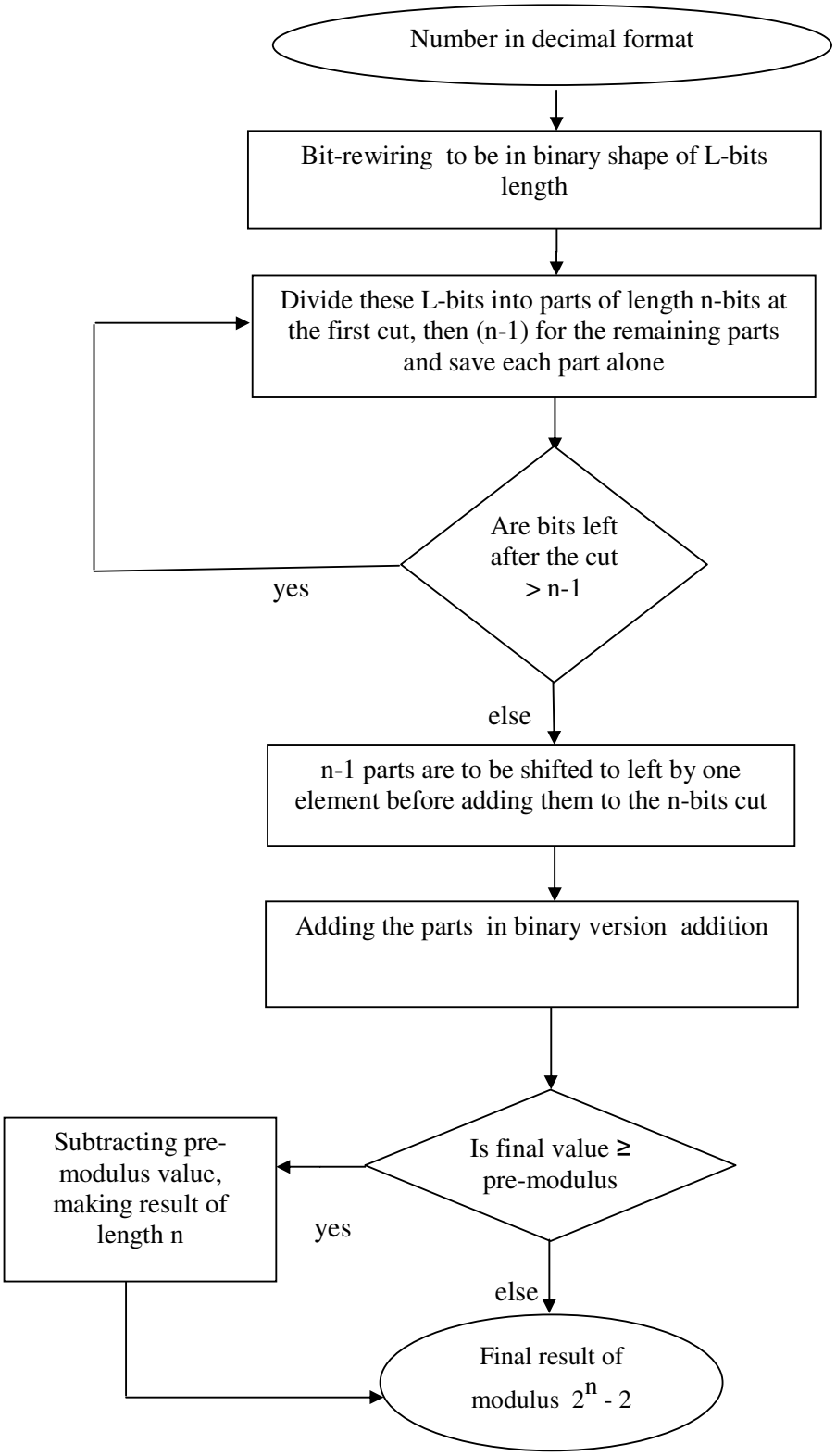


Figure 1. Computing the residue with respect to 2^n-2.

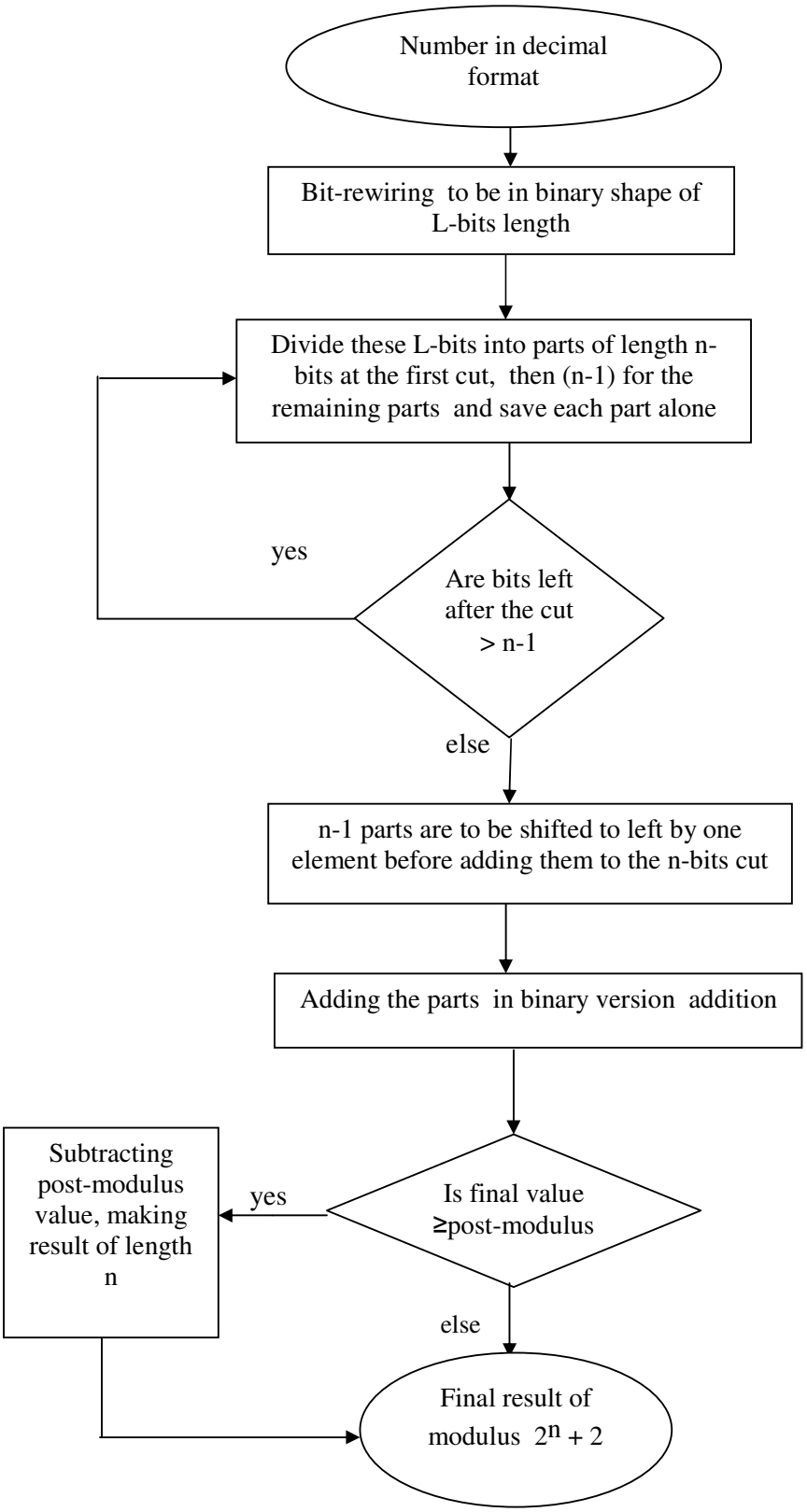


Figure 2. Computing the residue with respect to 2^n+2.

4. ALGORITH IMPLEMENTATION

In this section the implementation of the new non-coprime moduli set converter's implementation is done through hardware block diagrams. As described in the previous section, the converter has 3 stages of computing the modulus; they are (pre-modulus '$2^n - 2$', middle modulus '2^n' and post-modulus '$2^n + 2$'). So this section also consists of three hardware implementation blocks of them.

4.1 Pre-modulus Hardware Implementation

From the proposed algorithm presented for the pre-modulus calculation discussed in the flowchart of figure 1, we can come with this block diagram for the hardware design of it in figure 3.

Notice that the start for cutting is from LSB side to the MSB. The DR for any n value is calculated as was shown in formula (2), thus any number inside the range of M has 3 part cuts -at most-. For example let n = 5, M = 15*32*17 = 8160, so the numbers inside the DR are {0,1,2,……,8159}. However 8159 is represented in binary form as (1111111011111) which consist of 13 digits and it is equal to 5 + 4 + 4 as the cuts presented by its algorithm showed. This is true for all n values, so we will name the first n-bits cut A, the second (n-1) bits cut B and the final part of (n-1) bits is C.

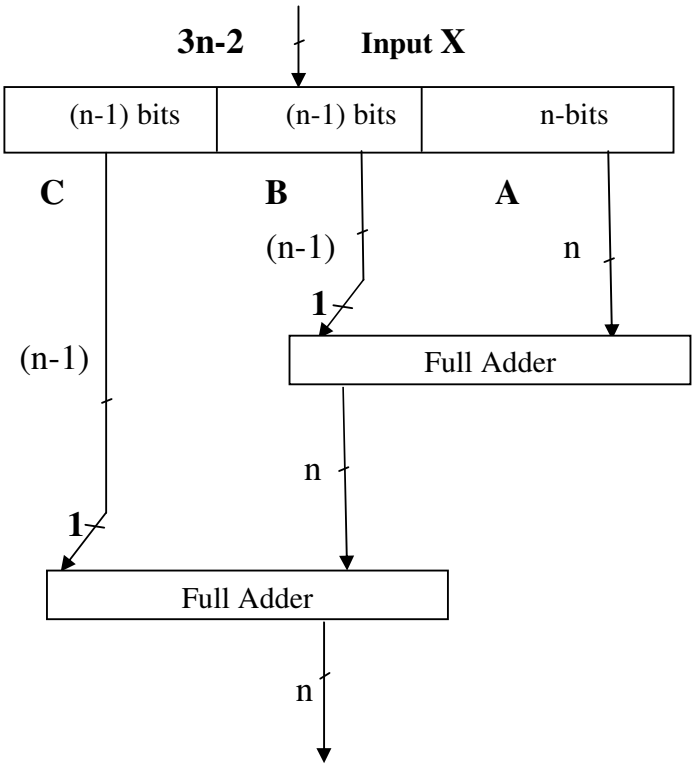


Figure 3. Calculation of Pre-modulus ($2^n - 2$).

4.2 Middle-modulus Hardware Implementation

No change is done on it, so we refer you to [15]-[17]. Obtaining the residue of X with respect to modulus 2^n is the easiest operation. Block diagram of it is shown in figure 4.

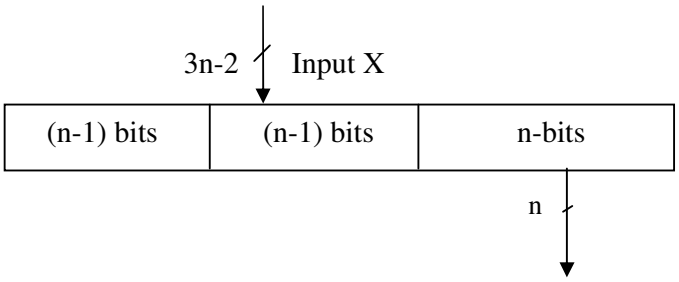


Figure 4. Calculation of Middle-modulus (2^n).

4.3 Post-modulus Hardware Implementation

From the proposed algorithm presented for the post-modulus ($2^n + 2$) calculation discussed in the flowchart of figure 2, we can come with this block diagram for the hardware design of it in figure 5.

Notice that the start for cutting is from LSB side to the MSB. The DR for any n value is calculated as was shown in formula (2), thus any number inside the range of M has 3 part cuts -at most-. For example let n = 5, M = 15*32*17 = 8160, so the numbers inside the DR are {0,1,2,……,8159}. However 8159 is represented in binary form as (1111111011111) which consists of 13 digits and it is equal to 5 + 4 + 4 as the cuts presented by its algorithm showed. This is true for all n values, so we will name the first n-bits cut A, the second (n-1) bits cut B and the final part of (n-1) bits is C.

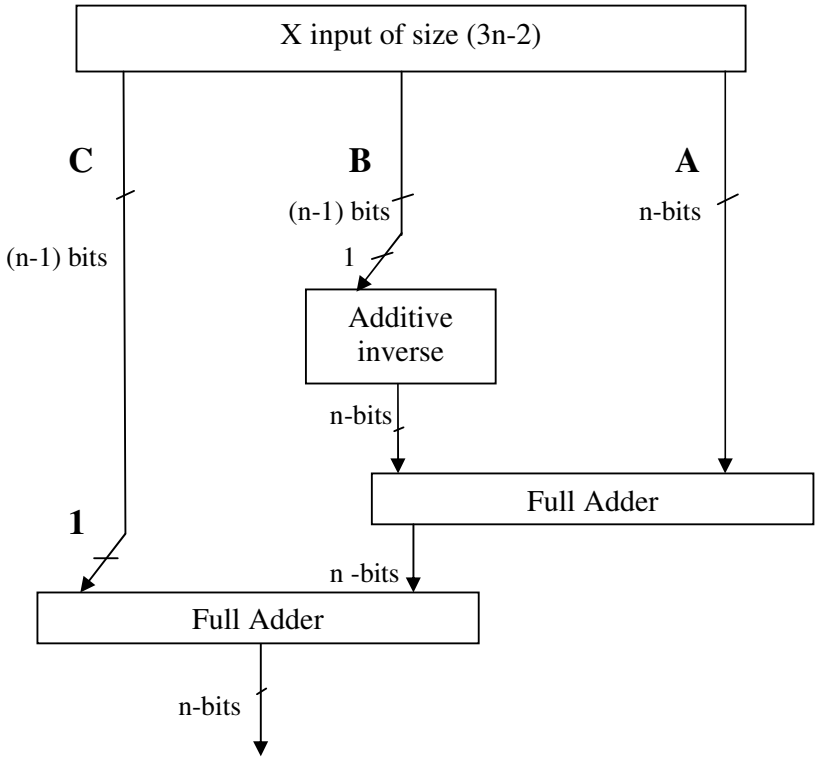


Figure 5. Calculation of Post-modulus ($2^n + 2$).

5. CONCLUSIONS

A new non-coprime moduli set has been proposed. A general formula for the dynamic range was derived. Algorithm of the special non-coprime moduli set has been suggested. The uniqueness for the new special non-coprime moduli set just as the co-prime one's among DR has been verified.

This research revealed that non-coprime moduli set may be suitable for wide variety of cases not limited to co-prime only.

ACKNOWLEDGEMENTS

The authors would like to thank everyone.

REFERENCES

[1] Neha Singh, (2008), "An overview of Residue Number System". National Seminar on Devices, Circuits & Communication.

[2] Chaves, R., and Sousa, L. (2007) "Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures". Computers &Digital Techniques IET, Volume 1, Issue 5, pages 472-480.

[3] Modiri, Samira, Movaghar, and Barati (2012) "Study of error control capability for the new moduli set {22n+ 1+ 2n-1, 22n+ 1-1, 2n-1, 23n, 23n+ 1-1}". Journal of Advanced Computer Science & Technology, Vol. 1, Pages: 176-186.

[4] Abdelfattah (2011) “Data Conversion in Residue Number System”. McGill University.

[5] Bajard and Plantard(2004) "RNS bases and conversions". Proceedings of Advanced Signal Processing Algorithms, Architectures, and Implementations XIV SPIE, Vol. 5559,Page:61-75.

[6] William A. Chren, Jr., (2005), "Residue Number System Arithmetic Circuits With Built-in Self Test". United States Patent 6886123 B1.

[7] http://homepage.cs.uiowa.edu /~jones/bcd/mod.shtml, last accessed on October, 2015.

[8] https://en.wikipedia.org/wiki/Mersenne_prime, last accessed on October, 2015.

[9] Y. Wang.(1998) "New Chinese Remainder Theorems". In proceedings of the Thirty Second Asilomar Conference on Signals, Systems and Computers, Pages: 165-171.

[10] Vidhyalakshmi.M,Prof.Satyabama . (2014) “Design and Implementation of Efficient Binary to Residue Converter Using Moduli Method”. International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Special Issue 1, March 2014.

[11] Shende, Radha, and Zode.(2012) "Efficient design 2k– 1 binary to residue converter." On proceedings of International Conference IEEE, Devices, Circuits and Systems (ICDCS), Pages: 482 – 485.

[12] Omondi, Amos, and Premkuar.(2007) “Residue number systems: theory and implementation”. Imperial College Press.

[13] Mohan,and Ananda(2002) "Residue number systems: algorithms and architectures". Springer, vol.1, pages:1-268.

[14] Jameii, Mahdi, Taghipour, and Azad (2011) "Using both Binary and Residue Representations for Achieving Fast Converters in RNS".Journal of advances in computer research, Pages: 91-104.

[15] Hiasat, and Sweidan(2003) "Residue number system to binary converter for the moduli set (2n– 1, 2n, 2 n+ 1)". Journal of systems architecture,Vol.49, Pages: 53-58.

[16] Bajard and Plantard(2004) "RNS bases and conversions". Proceedings of Advanced Signal Processing Algorithms, Architectures, and Implementations XIV SPIE, Vol. 5559,Page:61-75.

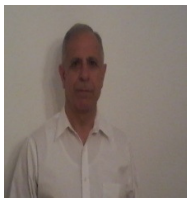
[17] Ricardo Chaves, and Leonel Sousa, (2004), "{2n + 1, 2n+k, 2n – 1} : A New RNS Moduli Set Extension". Euromicro Symposium Conference: Digital System Design.

[18] Taleshmekaeil, D.K., and Mousavi, A. (2010), "The use of Residue Number System for improving the Digital Image Processing". IEEE 10th International Conference on Signal Processing (ICSP), pages 775-780.

[19] Neha Singh, (2008), "An overview of Residue Number System". National Seminar on Devices, Circuits & Communication.

AUTHORS

Mansour Bader holds a MSc in computer engineering and networks, University of Jordan, Jordan, 2016. BSc Computer Engineering, Al-Balqa Applied University, Jordan, 2008. He is a technical support engineer of computer networks at computer center of Al-Balqa Applied University for 8 years and a half.



Dr. Andraws I. Swidan was born in Al-Karak Jordan in 1954. He received his diploma in Computer Engineering (with honours) and Ph.D. in Computer Engineering from LETI Ulianov Lenin, Sanct Peterburg (Leningrad), Russia in 1979 and 1982 respectively. He Joined the Electrical Engineering Department at the University of Jordan in 1983 and was one of the founders of the Computer Engineering Department at the University of Jordan in 1999. Since then he is a professor at the department. He is also an Adjunct Professor with the ECE department of the McGill University, Montreal, Canada. He holds several technical certifications among which the CISSP. He is an IEEE member, Jordanian Engineering Association member Quebec College of engineers member. He is a Canada Professional Engineer (The province of Quebec). He was member of several national and international scientific committees. He holds several patents and tens of publications. His main areas of research interest are: computer arithmetic, computer security, encryption algorithms.

Mazin Al-hadidi has PhD. in Engineering Science (Computing Machines, Systems and Networks), Institute of Problems of Simulation in Power Engineering Academy of Science, Ukraine/Kiev .1990-1994, with grade Excellent. Bachelor and Master Degrees in Engineering (Computer and intellectual systems and networks) Kiev Institute of Civil Aviation Engineers, as a governmental scholarship, Soviet Union / Kiev, 1984-1990, with grade very good. General Secondary 12 Years Certificate in the Science branch, Jordan/Al-Salt, 1984, with grade very good.

Baha Rababah has a MSc of computer network administration and management (with Distinction), University of Portsmouth, UK, 2015. BSc Computer Engineering, Al-Balqa Applied University, Jordan, 2010. He is a lecturer of computer networks subjects in Keys Training and Solution, Irbid, Jordan.