# An $O(n)$ Residue Number System to Mixed Radix Conversion Technique

Kazeem Alagbe Gbolagade[1,2], Member, IEEE and Sorin Dan Cotofana[1], Senior Member IEEE,
1. Computer Engineering Laboratory, Delft University of Technology,
The Netherlands. E-mail: {gbolagade,sorin}@ce.et.tudelft.nl
2. University for Development Studies, Navrongo, Ghana.

***Abstract*—This paper investigates the conversion of Residue Number System (RNS) operands to decimal, which is an important issue concerning the utilization of RNS numbers in digital signal processing applications. In this line of reasoning, we introduce an RNS to Mixed Radix Conversion (MRC) technique, which addresses the computation of Mixed Radix (MR) digits in such a way that enables the MRC parallelization. Given an RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$, the key idea behind the proposed technique is to maximize the utilization of the modulo-$m_i$ adders and multipliers present in the RNS processor functional units. For an $n$-digit RNS number $X = (x_1, x_2, x_3, ..., x_n)$ the method requires $n$ iterations. However, at iteration $i$, the modulo-$m_i$ units are utilized for the calculation of the MR digit $a_i$, while the other modulo units are calculating intermediate results required in further iterations. Our approach results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while state of the art MRCs exhibit an asymptotic complexity in the order of $O\left(n^2\right)$. More in particular, when compared with the best state of the art MRC, our technique reduces the number of arithmetic operations by $5.26\%$ and $38.64\%$ for moduli set of length four and ten, respectively.**

***Index Terms*—Residue Number System, Mixed Radix Conversion, Data Conversion, Mixed Radix Digits, arithmetic operations.**

## I. INTRODUCTION

Residue Number System $(RNS)$ [1], [2] is an integer number system with the capabilities to support parallel, carry-free addition, borrow-free subtraction and single step multiplication without partial product. These features enable RNS utilization in Digital Signal Processing (DSP) applications such as digital filtering, convolution, fast Fourier transform and image processing [13], [15]. For successful application of RNS, data conversion must be very fast so that the conversion overhead doesn't nullify the RNS advantages [16].

The work on residue to binary conversion is based on Chinese Remainder Theorem (CRT) [7]-[9],[11],[15],[16] or on Mixed Radix Conversion (MRC) [3]-[6],[10],[12],[14]. CRT is desirable because the computation can be parallelized while MRC is by its very nature a sequential process. However many up to date RNS to binary/decimal converters are based on MRC due to the complex and slow modulo-M operation (M being the system dynamic range thus a rather large constant) required by CRT. The main problem with the MRC is that the computations of the MR digits is done in a serial manner and requires a large number of arithmetic operations.

In this paper, we introduce an RNS to MRC technique, which addresses the computation of Mixed Radix (MR) digits in such a way that enables the MRC parallelization. Our approach results in an RNS to MRC with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while state of the art MRCs exhibit an asymptotic complexity in the order of $O\left(n^2\right)$. More in particular, when compared with the state of the art MRC in [14], our technique reduces the number of arithmetic operations by $5.26\%$ and $38.64\%$ for moduli set of length four and ten, respectively.

The rest of this paper is organized as follows: First we briefly present the necessary background in Section II. In Section III we introduce our Mixed Radix Conversion technique. The performance of our proposal is evaluated in Section IV while some conclusions are drawn in Section V.

## II. BACKGROUND

RNS is defined in terms of a set of relatively prime integers $\{m_i\}_{i=1,n}$ such that $gcd\,(m_i, m_j) = 1$ for $i \neq j$, where $gcd$ means the greatest common divisor of $m_i$ and $m_j$. For such a system $M = \Pi_{i=1}^{n} m_i$, is the dynamic range and any integer $X \in [0, M-1]$ can be uniquely represented as $X = (x_1, x_2, x_3, ..., x_n)$, where $x_i = |X|_{m_i}$, $0 \leq x_i < m_i$. We note here that in this paper we use $|X|_{m_i}$ to denote the $X$ mod $m_i$ operation and the operator $\Theta$ to represent the operation of addition, subtraction, and multiplication. Given any two integer numbers $K$ and $L$ in RNS represented by $K = (k_1, k_2, k_3, ..., k_n)$ and $L = (l_1, l_2, l_3, ..., l_n)$, respectively, $W = K \Theta L$, can be calculated as $W = (w_1, w_2, w_3, ..., w_n)$, where $w_i = |k_i \Theta l_i|_{m_i}$, for $i = 1, n$. This actually means that the complexity of the calculation of the $\Theta$ operation is determined by the number of bits required to represent the residues and not by the one required to represent the input operands.

The conversion from RNS to decimal using MRC can be formulated as follows [2]:

Given an $n$-digit number $X = (x_1, x_2, x_3, ..., x_n)$ in an RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$ find a set of digits $\{a_1, a_2, a_3, ..., a_n\}$, which are the mixed radix digits (MRDs), such that Equation (1) holds true.

$$\begin{aligned} X &= a_1 + a_2 m_1 + a_3 m_1 m_2 + ... \\ &\quad + a_n m_1 m_2 m_3 ... m_{n-1} \end{aligned} \tag{1}$$

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

The mixed radix digits can be computed as follows [14]:

$$\begin{aligned} a_1 &= x_1 \\ a_2 &= \left|(x_2 - a_1)\left|m_1^{-1}\right|_{m_2}\right|_{m_2} \\ a_3 &= \left|\left((x_3 - a_1)\left|m_1^{-1}\right|_{m_3} - a_2\right)\left|m_2^{-1}\right|_{m_3}\right|_{m_3} \\ &\ldots \\ a_n &= |((...(x_n - a_1)|m_1^{-1}|_{m_n} - a_2)|m_2^{-1}|_{m_n} - ... \\ &\quad -a_{n-1})|m_{n-1}^{-1}|_{m_n}|_{m_n} \end{aligned} \tag{2}$$

Given the MRD $a_i, 0 \leq a_i < m_i$, any positive number in the interval [0, $\Pi_{i=1}^{N} m_i - 1$] can be uniquely represented. One can easily deduce from Equation (2) and also in line with the discussion in [14] that a total of $\frac{n(n-1)}{2}$ arithmetic subtractions and multiplications are required. This means that the conversion process that computes the MRDs for an RNS with an $n$-moduli set has an asymptotic complexity in the order of $O(n^2)$. In the general case, due to the fact that all the $m_i$ products in Equation (1) can be precalculated, $n-1$ multiplications and $n-1$ additions are required for the conversion of an MR number to binary/decimal. That part of the calculation cannot be diminished as it stands on the very nature of MR representation.

The improved MRC described in [14] requires a total of $\frac{n(n-1)}{2}$ subtractions just as in [2] but can reduce the arithmetic multiplications to $n-2$ instead of $\frac{n(n-1)}{2}$ required in [2] for computing the MRDs. Computation of the MRDs in a faster way based on look up tables has been described in [3]-[6]. The complexity of the MRC described in [2], [3]-[6], and [14] either in terms of the number of arithmetic operations or in terms of the required number of look-up tables is in the order of $O(n^2)$. The algorithm in [6] is reported to be better than that in [3] and [4]. The algorithm presented in [6] is rather complex as it requires solving $\frac{n(n-1)}{2}$ linear Diophantine Equations (DE). The algorithm presented in this paper follows in a certain way the same assumptions and principles as [6] but without the need to solve any DE.

Our proposal is stemming from the fact that in Equation (2) not all the ways in the modulo-$m_i$ functional units are utilized at each iteration. Given the fact that in one RNS operation all the modulo-$m_i$ units can execute useful computation the conversion algorithm which directly follows Equation (2) is under-utilizing the available hardware in the RNS processor. Based on this observation and in a more simpler manner when compared to [6], we present in the next section a new method to compute the MR digits $a_i$ that exhibits more parallelism and entirely utilizes the modulo-$m_i$ ways in the RNS processor functional units.

## III. $O(n)$ MIXED RADIX CONVERSION TECHNIQUE

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

Suppose we have a set of residues $\{x_1, x_2, x_3, ..., x_n\}$ corresponding to the set of moduli $\{m_1, m_2, m_3, ..., m_n\}$, then the mixed radix digits $a_i$ and the decimal equivalent can be computed using Equations (2) and (1). As previously mentioned, the challenge is to obtain the MRDs $a_i$ in a more parallel manner. All the derivations in this section are done under the assumption that $1 < m_1 < m_2 < m_3 < ... < m_n$ holds true.

In Equation (1), every term except the last, i.e, $a_1$, is a multiple of $m_1$. If we take the modulo of both sides of Equation (1) with respect to $m_1$, we obtain:

$$|X|_{m_1} = a_1 \tag{3}$$

meaning that $a_1 = x_1$.

In a similar manner, if we subtract $a_1$ from both sides of Equation (1), divide both sides by $m_1$, and then compute the modulo of both sides with respect to $m_2$ we obtain:

$$a_2 = \left|\frac{X - a_1}{m_1}\right|_{m_2}, \tag{4}$$

which is equivalent to:

$$a_2 = \left|\left|(m_1)^{-1}\right|_{m_2} |(x_2 - x_1)|_{m_2}\right|_{m_2}. \tag{5}$$

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

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

**Level 0**: The first level is regarded as level 0 and in this level no calculations are required as indicated by Equation (6). This implies that $n-1$ levels are required for a system involving $n$-digit MR conversion.

$$y_1^0 = a_1 \quad \text{and} \quad a_1 = x_1 \tag{6}$$

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

$$y_{j+1}^1 = \left|\frac{x_{j+1} - x_1}{m_1}\right|_{m_{j+1}}, \tag{7}$$

which implies that:

$$y_{j+1}^1 = \left| \left| m_1^{-1} \right|_{m_{j+1}} \left| (x_{j+1} - x_1) \right|_{m_{j+1}} \right|_{m_{j+1}} \quad (8)$$

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

$$j = 1, a_2 = y_2^1 = \left| \frac{x_2 - x_1}{m_1} \right|_{m_2}, \quad (9)$$

which implies that:

$$y_2^1 = \left| \left| m_1^{-1} \right|_{m_2} \left| (x_2 - x_1) \right|_{m_2} \right|_{m_2} \quad (10)$$

$$j = 2, y_3^1 = \left| \left| m_1^{-1} \right|_{m_3} \left| (x_3 - x_1) \right|_{m_3} \right|_{m_3} \quad (11)$$

$$j = 3, y_4^1 = \left| \left| m_1^{-1} \right|_{m_4} \left| (x_4 - x_1) \right|_{m_4} \right|_{m_4} \quad (12)$$

...

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

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

$$y_{j+2}^2 = \left| \left| m_2^{-1} \right|_{m_{j+2}} \left| \left( y_{j+2}^1 - y_2^1 \right) \right|_{m_{j+2}} \right|_{m_{j+2}}. \quad (13)$$

When $j = 1$, we have:

$$a_3 = y_3^2 = \left| \left| m_2^{-1} \right|_{m_3} \left| \left( y_3^1 - y_2^1 \right) \right|_{m_3} \right|_{m_3}. \quad (14)$$

When $j = 2$, we have:

$$y_4^2 = \left| \left| m_2^{-1} \right|_{m_4} \left| \left( y_4^1 - y_2^1 \right) \right|_{m_4} \right|_{m_4}. \quad (15)$$

When $j = 3$, we have:

$$y_5^2 = \left| \left| m_2^{-1} \right|_{m_5} \left| \left( y_5^1 - y_2^1 \right) \right|_{m_5} \right|_{m_5}. \quad (16)$$

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

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

$$y_{j+3}^3 = \left| \left| m_3^{-1} \right|_{m_{j+3}} \left| \left( y_{j+3}^2 - y_3^2 \right) \right|_{m_{j+3}} \right|_{m_{j+3}}. \quad (17)$$

When $j = 1$,

$$a_4 = y_4^3 = \left| \left| m_3^{-1} \right|_{m_4} \left| \left( y_4^2 - y_3^2 \right) \right|_{m_4} \right|_{m_4}. \quad (18)$$

When $j = 2$,

$$y_5^3 = \left| \left| m_3^{-1} \right|_{m_5} \left| \left( y_5^2 - y_3^2 \right) \right|_{m_5} \right|_{m_5}. \quad (19)$$

When $j = 3$,

$$y_6^3 = \left| \left| m_3^{-1} \right|_{m_6} \left| \left( y_6^2 - y_3^2 \right) \right|_{m_6} \right|_{m_6}. \quad (20)$$

$y_4^3$, $y_5^3$, $y_6^3$,...,$y_n^3$ will be used in the next level and can be computed in parallel.

The rest of the MRDs are computed in a similar manner and the iteration continues until $a_n$ is computed as $y_{j+(n-1)}^{n-1}$ and

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

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

## IV. PERFORMANCE EVALUATION

When analyzing the proposed method, one can observe the following: (i) no computation is required at Level 0; (ii) one computation is required in the last level, i.e, the computation of $a_n$; (iii) each of the remaining $(n-2)$ levels requires $((n-k), k = 1, 2, ..., n-1)$ computations, where by computation we mean one modulo addition and one modulo multiplication. One can observe however that the $(n-k)$ per level computations can be done in parallel as they utilize different modulo-$m_i$ ways in the RNS processor functional units. Given that they are equivalent to one computation in the RNS hardware. This implies that the total number of computations that are required in all the levels sums up to $n-1$. Hence, the asymptotic complexity of the proposed technique is in the order of $O(n)$. This constitutes a substantial improvement over the state of the art as the MRCs described in [3]-[6], either in terms of the number of arithmetic operations or in terms of the required number of look-up tables, have asymptotic complexities in the order of $O(n^2)$.

In order to get a better estimate of the impact of our method in practice, we compute the number of required operations for the classic MRC, the method in [14] (MRC14), and our method (IMRC), for moduli sets of length of 3 to 10. The total number of operations is computed based on the assumption that an addition takes one cycle and a multiplication two cycles, thus we consider that one multiplication is equivalent delay wise with two additions. MRC and MRC14 require $\frac{n(n-1)}{2}$ and $(n-2)$ multiplications, respectively, and the same $\frac{n(n-1)}{2}$ additions for the computation of the MRDs. Additionally, all the methods require $(n-1)$ additions and $(n-1)$ multiplications to compute the decimal number according to Equation (1). As the moduli set cardinality increases, the number of arithmetic operations in the traditional MRC grows quadratically while for IMRC and the MRC14, it increases with a constant factor of 6 and $((8+k), k = 0, 1, 2, ..)$ arithmetic operations, respectively. Thus MRC14 also increases quadratically.

The results of this comparison are depicted in Table I and Figure I. Table I presents the percentage reduction of the total number of arithmetic operations required by IMRC when compared to MRC and MRC14. We note here that in Table I, the following notations are utilized: Mod- stands for the number of moduli in the considered RNS; RI- stands for reduction of the total number of arithmetic operations in percentage achieved by IMRC over classical MRC; while RII- stands for reduction of the total number of arithmetic operations in percentage achieved by IMRC over MRC14. One can observe that IMRC achieves $33.33\%$ and $5.26\%$ reductions

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

Table I
ARITHMETIC OPERATIONS REDUCTION IN %

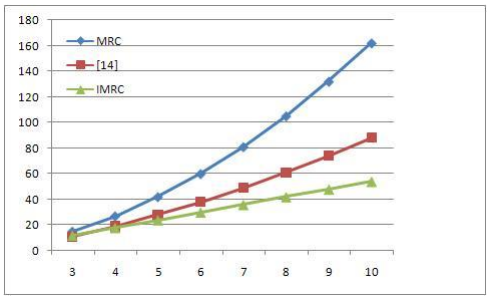


Figure 1. Number of arithmetic operations Vs Moduli Set Length

with moduli sets of length four when compared with the classic MRC and MRC14, respectively. For moduli sets of length ten, IMRC achieves $66.67\%$ and $38.64\%$ reductions when compared with the classic MRC and MRC14, respectively. As expected, the larger the number of moduli in the RNS, the larger the reduction the proposed conversion method exhibits. The traditional MRC and the one in [14] respectively require $\frac{n(n-1)}{2}$ and $n-2$ multiplications with the same $\frac{n(n-1)}{2}$ additions for the computation of MRDs in addition to $n-1$ additions and $n-1$ multiplications required by Equation (1) to compute the decimal number. However, it should be noted that it is not in every case that the improved MRC in [14] reduces the arithmetic multiplications to $n-2$. We thus compare our proposal with this maximum arithmetic multiplication that can be provided by [14].

Figure I depicts the IMRC, MRC14, and the classic MRC performance in terms of the number of arithmetic operations as the length of the moduli set increases. Clearly, it can be seen from Figure I that our proposal has the least growth in the RNS arithmetic operations as moduli set length increases when compared with MRC and MRC14.

## V. CONCLUSIONS

In this paper, we investigated the conversion of RNS operands to binary/decimal, which is an important issue that enables/precludes the utilization of RNS numbers in addition and multiplication dominated DSP applications. We introduced an RNS to MRC technique, which addresses the computation of MR digits in such a way that enables the MRC parallelization. Given an RNS with the set of relatively prime integer moduli $\{m_i\}_{i=1,n}$, the key idea behind the proposed technique is to maximize the utilization of the modulo-$m_i$ adders and multipliers present in the RNS processor functional units. For an $n$-digit RNS number $X = (x_1, x_2, x_3, ..., x_n)$ the method requires $n$ iterations. However, at iteration $i$, the modulo-$m_i$ units are utilized for the calculation of the MR digit $a_i$, while the other modulo units are calculating intermediate results required in further iterations. Given that one such iteration can be seen as one single operation on the RNS processor functional units, our approach results in an RNS to MR conversion with an asymptotic complexity, in terms of arithmetic operations, in the order of $O(n)$, while the traditional MRC and many other state of the art MRCs based techniques exhibit an asymptotic complexity in the order of $O\left(n^2\right)$. More in particular, the utilization of our technique achieved $33.33\%$ and $5.26\%$ reductions with moduli sets of length four when compared with MRC and the improved MRC in [14],respectively. For moduli sets of length ten, our proposal achieved $66.67\%$ and $38.64\%$ reductions when compared with MRC and the improved MRC in [14], respectively. Given that the method we proposed substantially reduces the RNS to binary/decimal conversion overhead it potentially makes RNS more effective in addition and multiplication dominated DSP applications.

## REFERENCES


[1] H.L. Garner, The residue Number System, IRE Trans. on Electronic Computers, pp. 140-147, 1959.

[2] Szabo, N., and Tanaka, R., Residue arithmetic and its application to computer technology, McGraw-Hill, New York, 1967.

[3] N.B. Chakraborti, J.S. Soundararajan and A.L.N. Reddy, An implementation of mixed-radix conversion for residue number applications, IEEE Trans. Computers, Vol. C-35, Aug., 1986.

[4] C.H. Huang, A fully parallel mixed-radix conversion algorithm for residue number applications, IEEE Trans. Computers, Vol. C-32, pp. 398-402, April, 1983.

[5] G.A. Jullien, Residue Number Scaling and other Operations using ROM arrays, IEEE Trans. Computers, Vol. C-27, pp. 325-336, April, 1978.

[6] D.F. Miller and W.S. McCormick, An arithmetic free parallel mixed-radix conversion algorithm, IEEE Trans. Circuits Syst. II Analog and Digital Signal Processing, Vol. 45, pp. 158-162, Jan., 1998.

[7] M.O. Ahmad, Y. Wang, M.N.S Swamy, Residue to Binary Converters for three moduli set, IEEE Trans. Circuits Syst. II,, Vol. 46, pp. 180-183, Feb., 1999.

[8] W. Wang, M.N.S. Swamy, M.O. Ahamad and Y. Wang: A high - Speed residue-to-binary converter and a Scheme for its VLSI Implementation. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 330-333, 1999.

[9] W. Wang, M.N.S. Swamy, and M.O. Ahmad, An Area-Time efficient residue-to-binary converter. 43rd IEEE MidWest Symp. On Circuits and Systems, Lansing MI, Aug.8-11,2000.

[10] H.M. Yassine, Fast Arithmetic based on Residue Number System Architectures. IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, pp. 2947-2950, 1991.

[11] A.A. Hiasat, Efficient Residue to Binary Converter. IEE Proceedings. Computers and Digital Techniques (IET Research Journals), Vol. 150, Issue 1, pp. 11-16, 2003.

[12] H.M. Yassine, Matrix Mixed-Radix Conversion For RNS Arithmetic Architectures. 34th Midwest Symposium on Circuits and Systems, pages 273-278,1992.

[13] W.Jenkins and B. Leon, The use of residue number systems in the design of finite impulse response digital filters. IEEE Transaction on Circuits and Systems, Vol.24, pp191-201, April, 1987.

[14] H.M. Yassine and W.R. Moore, Improved mixed-radix conversion for residue number architectures, IEE Proceedings-G, Vol. 138, No.1 pp. 120-124, Feb. 1991.

[15] K.A. Gbolagade and S.D. Cotofana,Residue Number System Operands to Decimal Conversion for 3-moduli sets, 51st Midwest Symposium on Circuits and Systems, Knoxville, USA, pp. 791-794, August, 2008.

[16] K.A. Gbolagade and S.D. Cotofana,A Residue to Binary Converter for $\{2n+2, 2n+1, 2n\}$ Moduli Set, 42nd Asilomar Conference on Circuits and Systems, California, USA, October 26-29, 2008 (to appear).