Sādhanā (2019) 44:99
https://doi.org/10.1007/s12046-019-1078-0

Novel RNS-to-binary converters for the three-moduli set $\{2m-1, 2m, 2m+1\}$

P S PHALGUNA[1], DATTAGURU V KAMAT[1] and P V ANANDA MOHAN[2,*]

[1]Department of Electronics and Communication Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal 576 104, India
[2]R&D, Centre for Development of Advanced Computing, 1, Knowledge Park, Bayappanahalli, Bangalore 560 038, India
e-mail: phalguna.ps@learner.manipal.edu; dv.kamath@manipal.edu; anandmohanpv@live.in

MS received 14 December 2017; revised 23 August 2018; accepted 11 January 2019; published online 27 March 2019

Abstract. In this paper, Mixed Radix Conversion (MRC)-based Residue Number System (RNS)-to-binary converters for the three-moduli set $\{2m-1, 2m, 2m+1\}$ are presented. The proposed reverse converters are evaluated and compared to reverse converters proposed earlier in literature using Chinese Remainder Theorem (CRT) and New CRT for this moduli set as well as two four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$ regarding hardware requirement and conversion time.

Keywords. Residue Number System; reverse converter; three-moduli set; CRT; Mixed Radix Conversion.

1. Introduction

The advantages of Residue Number System (RNS) such as carry-free operation, modularity and fault tolerance have made it attractive in applications like cryptography, digital signal processing (DSP) and communication systems [1–4]. Several three-, four- or more-moduli sets have been described in literature. They use powers-of-two-related moduli of the form 2^u, 2^u+1, 2^u-1, 2^v+3, 2^v-3. In addition, other three-moduli sets that use consecutive numbers as moduli also have been investigated, viz., $\{2m-1, 2m, 2m+1\}$ [5] and $\{2m, 2m+1, 2m+2\}$ [6], the latter using two moduli that have a common factor. The moduli sets $\{2^\alpha-1, 2^\alpha, 2^\alpha+1\}$ [7–12] and $\{2^{\beta-1}-1, 2^\beta-1, 2^\beta\}$ [13–16] are special cases of these two-moduli sets. Note that the moduli set $\{2^{\beta-1}-1, 2^\beta-1, 2^\beta\}$ is obtained by removing the common factor from one of the two even moduli $2^\beta-2$ and 2^β in the moduli set $\{2^\beta-2, 2^\beta-1, 2^\beta\}$ to make the moduli relatively prime. The moduli set $\{2^\alpha-1, 2^{\alpha+\gamma}, 2^\alpha+1\}$ has been also investigated to give a variable dynamic range (DR) using the additional degree of freedom γ where $0 \le \gamma \le \alpha$. [17]. This gives an increment of DR by γ bits over the moduli set $\{2^\alpha-1, 2^\alpha, 2^\alpha+1\}$ with a resolution of 1 bit. On the other hand, the moduli set $\{2m-1, 2m, 2m+1\}$ also offers several other options for realizing a desired DR through proper choice of m. As an illustration, the DRs of the popular moduli set starting from $\alpha = 3$, 4 and 5 are, respectively, 504, 4080 and 32736. The choice of variable γ leads to the DRs that are 1008, 2016, 4032, etc. In the case of $\{2m-1, 2m, 2m+1\}$ starting from $m = 3, 4, 5, 6, 7,$ etc. the DRs are 210, 504, 990, 1716, 2730, 5814, 7980, etc.

Premkumar [5] suggested the three-moduli set M1 $\{2m-1, 2m, 2m+1\}$ and several reverse converters for M1 have been reported in the literature [5, 18–21]. The first reverse converter for M1 is presented in [5] using CRT. Later, two reverse converters were presented using a modification of CRT for reducing the modulo reduction complexity [18]. Reverse converters for this moduli set using New CRT II [22] also have been investigated [19]. More recently, improved reverse converters for this moduli set using CRT have been presented [20, 21]. However, these converters can be considered to be similar to a Mixed Radix Conversion (MRC)-type design. The intermediate digits derived, however, are not amenable for facilitating comparison since one of the intermediate digits can be negative. It is interesting to note that MRC technique has not been explored for the moduli set M1. It is well known that MRC technique facilitates easy comparison of two RNS numbers as well as scaling by one modulus or product of two moduli [2]. In this paper, we consider the MRC technique for reverse conversion. Several architectures will be described that take advantage of the simper multiplicative inverses in order to arrive at designs with hardware requirement/conversion time trade-off. All the proposed architectures are compared to the state-of-the-art reverse converters reported earlier for the moduli set M1 in the literature as well as two representative four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$ regarding hardware requirement and conversion time.

*For correspondence

In section 2, background material has been given in brief. The proposed MRC-based reverse converter architectures are presented in section 3. The performance evaluation and comparison of the proposed converters with converters for M1 reported earlier and implementation results are provided in section 4. Comparison with converters for two representative four-moduli sets is also presented in section 4. The concluding remarks are given in section 5.

2. Background material

The two popular approaches used for the reverse conversion process in RNS are Chinese Remainder Theorem (CRT) and MRC. In CRT, we compute decoded binary number X as

$$X = \left(\sum_{i=1}^{j} x_i M_i \left(\frac{1}{M_i}\right)_{m_i}\right) \bmod M \tag{1}$$

where M is the product of all moduli m_i, $M_i = M/m_i$ and x_i are the given residues defined such that $x_i = X \bmod m_i$. Note that $y = \left(\frac{1}{a}\right)_b$ is known as a multiplicative inverse of a with respect to modulus b defined such that remainder of the computation $(a \times y)/b$ is 1. The main advantage of CRT is the parallel computation of various terms in (1) corresponding to the given residues followed by the summation of various terms mod M.

In MRC for three-moduli set $\{m_1, m_2, m_3\}$, the decoded number X corresponding to residues (x_1, x_2, x_3) is obtained as

$$X = U_3 m_2 m_1 + U_2 m_1 + U_1 \tag{2}$$

where the mixed radix digits U_i (i = 1, 2, 3) are computed as follows:

$$U_1 = x_1, \quad U_2 = \left((x_2 - x_1)\left(\frac{1}{m_1}\right)_{m_2}\right) \bmod m_2, \tag{3a}$$

$$U_3 = \left(\left(\left((x_3 - U_1)\left(\frac{1}{m_1}\right)_{m_3}\right)_{m_3} - U_2\right)\left(\frac{1}{m_2}\right)_{m_3}\right) \bmod m_3. \tag{3b}$$

Since MRC is a sequential process, in each step a single mixed radix digit is determined. The next step is to compute X in (2). Note that the cumbersome modulo M reduction needed in the case of CRT in (1) is not needed in MRC since $0 \le X < M$. In the present paper, we use MRC technique for deriving various reverse converters.

In the implementation of MRC, we need modulo subtractors for computing $(x_j - x_k) \bmod m_i$. They use structures similar to cost-effective (CE) and high-speed (HS) modulo adders [2]. We can use the HS architecture of figure 1a in which we compute $T_1 = (x_j - x_k)$ and $T_2 = (x_j - x_k + m_i)$ using two parallel adders and based on the sign of T_1 we select either T_1 or T_2 using a 2:1 multiplexer (2:1 MUX). Note that one's complement of x_k and a carry input of 1 are added to obtain two's complement of x_k. Thus the hardware requirement is two k-bit carry-propagate adders (CPA1 and CPA2), one k-bit carry-save adder (CSA1) and one k-bit 2:1 MUX where $k = \log_2(2m+1)$. The computation time is $(k+1)\Delta_{FA} + \Delta_{MUX}$ where Δ_{FA} and Δ_{MUX} are delays of a full adder and a 2:1 MUX, respectively. We denote this block as MODSUBA.

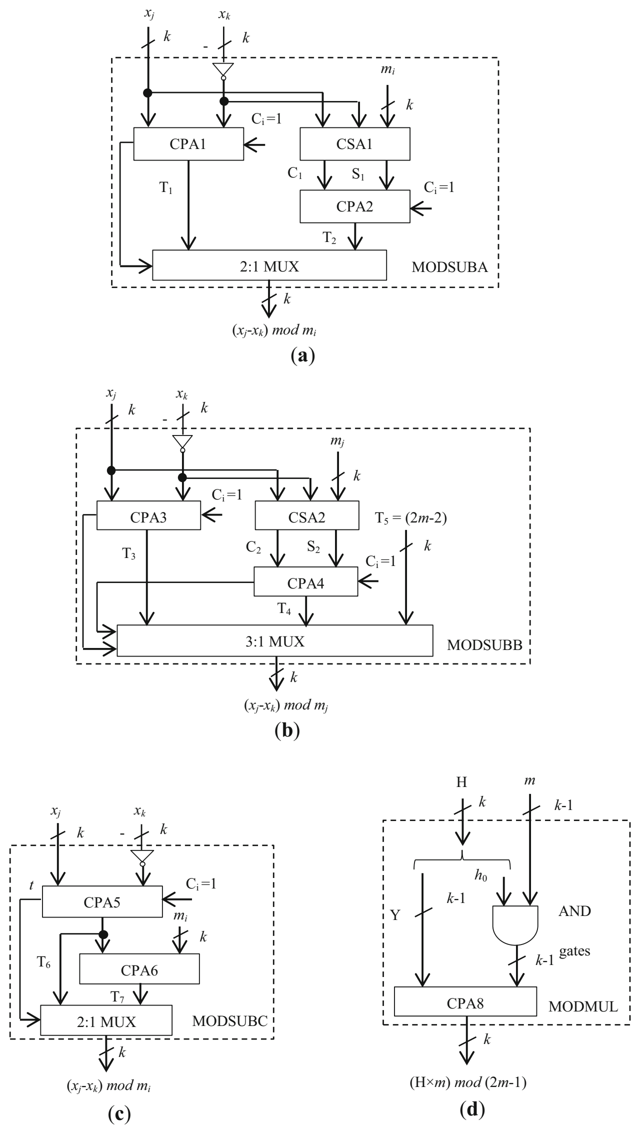


Figure 1. Architecture of (**a**) MODSUBA, (**b**) MODSUBB, (**c**) MODSUBC and (**d**) MODMUL (modulo multiplier $(H \times m) \bmod (2m-1)$).

Note, however, in the case of $m_j = 2m - 1$, $m_k = 2m + 1$, for computing $(x_j - x_k) \bmod m_j$ for $x_j = 0$, $x_k = 2m$, two consecutive additions of m_j are needed since $0 - 2m + (2m - 1) = -1$ and $-1 \bmod (2m-1)$ is $(2m - 2)$. Instead, we compute $T_3 = (x_j - x_k)$ or

$T_4 = (x_j - x_k + 2m - 1)$ and one among T_3, T_4 and $T_5 = 2m - 2$ can be selected using a 3:1 MUX based on the sign of T_3 and T_4 as shown in the MODSUBB block in figure 1b. The computation time, however, is about the same as that of MODSUBA.

The CE version of a modulo subtractor (MODSUBC block) can be realized as shown in figure 1c, in which we compute $T_6 = (x_j - x_k)$ followed by $T_7 = (T_6 + m_j)$ using two adders and based on the sign of T_6, we select either T_6 or T_7 using a 2:1 MUX. Thus the hardware requirement is two k-bit CPAs (CPA5 and CPA6) and one k-bit 2:1 MUX. The computation time needed is $(2k)\Delta_{\mathrm{FA}} + \Delta_{\mathrm{MUX}}$.

The implementation of $(H \times m)$ mod $(2m - 1)$ is also needed in the proposed reverse converter architectures. This can be carried out by considering $H = 2Y + h_0$, where Y is the word formed by $(k - 1)$-bit MSBs of the k-bit word H and h_0 is the LSB of H, as

$$\begin{aligned}(H \times m) \bmod (2m-1) &= (2Y \times m + h_0 \times m) \\ &\qquad \bmod (2m-1) \\ &= (Y + h_0 \times m) \bmod (2m-1).\end{aligned} \tag{4}$$

Note that Y is at most $H_{\max}/2 = (2m - 2)/2 = m - 1$, in which case $h_0 = 0$, thus making $Y + h_o \times m = m - 1$. In the other cases, $H_{\max}/2 < m - 1$ and even if $h_0 = 1$, $(Y + h_o m) \leq (m - 2) + m = 2m - 2 < 2m - 1$. Thus, $(H \times m)$ mod $(2m - 1)$ can be realized by adding Y with $h_o m$ (obtained by enabling m by h_o using $(k - 1)$ two-input AND gates) using CPA8 as shown in the MODMUL block of figure 1d. Note that m is available as $(k - 1)$ most significant bits of $m_2 = 2m$.

3. Proposed RNS-to-binary converters

In this section, we present new RNS-to-binary converters for the three-moduli set M1 $\{2m - 1, 2m, 2m + 1\}$ using MRC technique. The MRC algorithm for the three-moduli set M1 is shown in figure 2. The various multiplicative inverses needed in the computation are as follows:

$$a = \left(\frac{1}{2m+1}\right)_{2m} = 1, \tag{5a}$$

$$b = \left(\frac{1}{2m+1}\right)_{2m-1} = m, \tag{5b}$$

$$c = \left(\frac{1}{2m}\right)_{2m-1} = 1. \tag{5c}$$

They can be verified to be true since $(2m + 1) \times a = 1$ mod $2m$, $((2m + 1) \times b)$ mod $(2m - 1) = 1$ and $(2m) \times c = 1$ mod $(2m - 1)$. We denote the residues corresponding to the three-moduli $m_1 = 2m - 1$, $m_2 = 2m$ and $m_3 = 2m + 1$ as (x_1, x_2, x_3) and binary number corresponding to this residue set as X. The DR is

m_3 $= 2m+1$	m_2 $= 2m$	m_1 $= 2m$-1
x_3	x_2	x_1
	$\underline{-x_3}$	$\underline{-x_3}$
	$(x_2$-$x_3)$ *mod* $(2m)$ (= U$_A$*)	$(x_1$-$x_3)$ *mod* $(2m$-1) (= U$_B$*)
	$\underline{\times\ a\ (= 1)}$	$\underline{\times\ b\ (= m)}$
	U$_A$ (= (U$_A$*× a) *mod* $2m$)	U$_B$ (= (U$_B$*× b) *mod* $(2m$-1))
		$\underline{\text{-U}_A}$
		(U$_B$-U$_A$) *mod* $(2m$-1) (= U$_C$*)
		$\underline{\times\ c\ (= 1)}$
		U$_C$ (= (U$_C$*× c) *mod* $(2m$-1))

Figure 2. Conventional MRC for M1.

$M = 2m(4m^2 - 1)$. The implementation of the MRC algorithm of figure 2 using various multiplicative inverses Eq. (5a)–(5c) is presented in figure 3. This converter is denoted as D6.

The computation of $(x_2 - x_3)$ mod $2m$ can be carried out using CE version of a modulo subtractor MODSUBC of figure 1c to obtain intermediate result U_A*. The mixed radix digit U_A is thus already available as U_A* since a is 1 (see Eq. (5a)).

The computation of $U_B* = (x_1 - x_3)$ mod $(2m - 1)$ can be realized using MODSUBB block shown in figure 1b. Next, the intermediate result U_B is computed from U_B* by performing multiplication with b modulo $(2m - 1)$ in modulo multiplier block shown in figure 1d since b is m (see Eq. (5b)). Next, the modulo subtraction $(U_B - U_A)$ mod $(2m - 1)$ can be carried out using MODSUBA block to obtain U_C*. The mixed radix digit U_C is thus already available as U_C* since c is 1 (see Eq. (5c)).

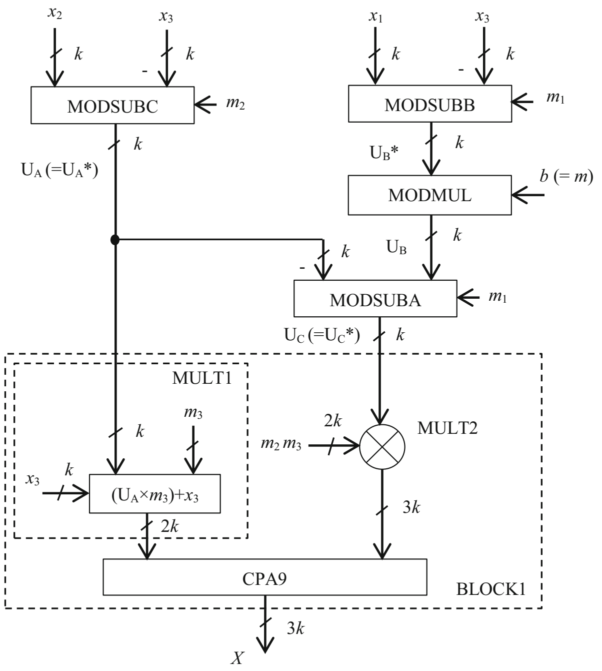


Figure 3. Architecture of MRC-based converter D6 for M1.

$m_2 = 2m$	$m_3 = 2m+1$	$m_1 = 2m-1$
x_2	x_3	x_1
	$\underline{-x_2}$	$\underline{-x_2}$
	(x_3-x_2)	(x_1-x_2)
	$\underline{\times\, e\ (=-1)}$	$\underline{\times\, f\ (=1)}$
	$P = (x_2-x_3+tm_3)$	(x_1-x_2)
		$\underline{-((x_2-x_3)+tm_3)}$
		$Q^*\ (=(x_1-2x_2+x_3-tm_3)\ mod\ m_1)$
		$\underline{\times\, g\ (=m)}$
		$Q\ (=(Q^*\times m)\ mod\ m_1)$

Figure 4. MRC for three-moduli set M1.

The last stage in the converter computes X using Eq. (2) as

$$X = (x_3 + U_A m_3) + U_C m_2 m_3. \tag{6}$$

Here the first term $(x_3 + U_A m_3)$ is computed using a $(k \times k)$-bit merged array multiplier MULT1 that multiplies two inputs U_A and m_3 and adds a third input x_3 in the carry save portion of the multiplier [23]. The second term $U_C m_2 m_3$ in Eq. (6) is computed using a $(2k \times k)$-bit array multiplier MULT2. Thus the decoded integer can be obtained using $3k$-bit CPA9 as shown in BLOCK1 of figure 3.

The design D6 is based on conventional MRC that requires sequential modulo reductions in the modulus m_1 channel to obtain the mixed radix digits. We explore techniques to reduce the number of cascaded modulo reductions next. For this purpose, we choose an ordering of moduli different from that shown in figure 2. The various multiplicative inverses needed for this approach shown in figure 4 are as follows:

$$e = \left(\frac{1}{2m}\right)_{2m+1} = -1, \tag{7a}$$

$$f = \left(\frac{1}{2m}\right)_{2m-1} = 1, \tag{7b}$$

$$g = \left(\frac{1}{2m+1}\right)_{2m-1} = m. \tag{7c}$$

The correctness of Eq. (7a)–(7c) can be easily verified.

The architecture of the converter D7 following figure 4 is shown in figure 5. The mixed radix digit P can be computed as $(x_2 - x_3) + tm_3$ since $e = -1$ (see Eq. (7a)) where if $x_2 \geq x_3$, t is 0, else t is 1. Note that $(x_2 - x_3)$-$+ tm_3$ is computed using MODSUBC block (see figure 5 with $x_j = x_2$ and $x_k = x_3$ and $m_i = m_3$). The sign bit of the result (the output of CPA5 in MODSUBC block in figure 1c) is considered as t.

Next, we consider computation of the mixed radix digit Q. We compute $(x_1 - x_2)$ but we defer modulo m_1 reduction since the multiplicative inverse f with which we need

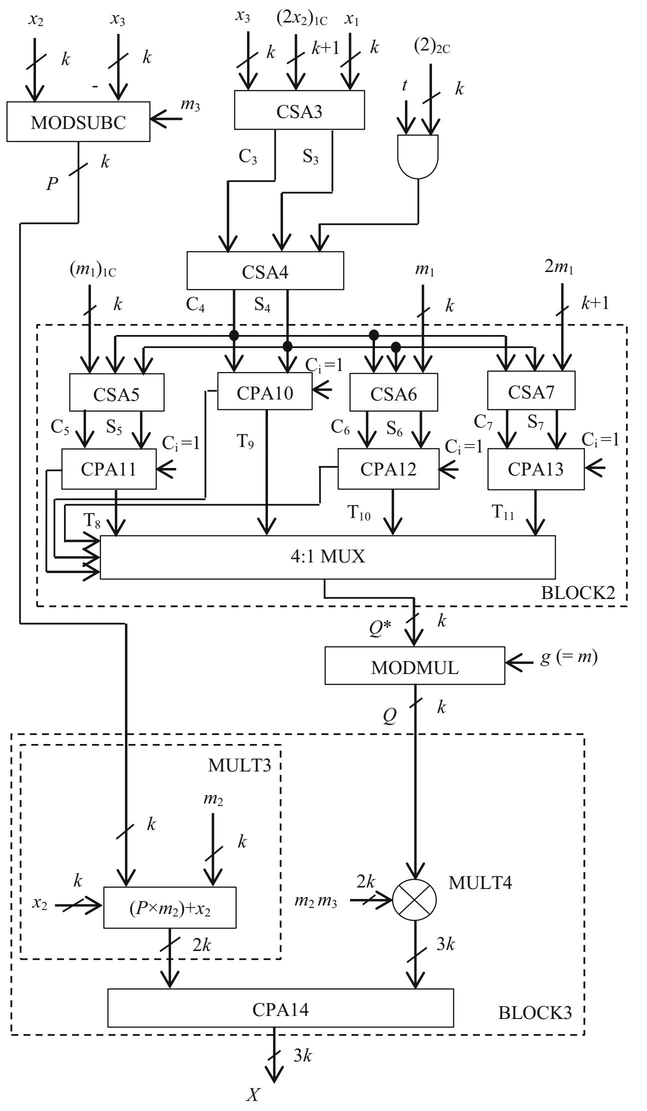


Figure 5. Architecture of the MRC-based converter D7 for M1.

to multiply mod m_1 is unity (see Eq. (7b)). Next, unlike in conventional MRC, we subtract $((x_2 - x_3) + tm_3)$ from $(x_1 - x_2)$ to obtain the intermediate result:

$$\begin{aligned} Q^* &= (x_3 - 2x_2 + x_1 - tm_3) \bmod m_1 \\ &= (x_3 - 2x_2 + x_1 - 2t) \bmod m_1. \end{aligned} \tag{8}$$

Note that in the second equality, we have used the fact m_3 mod $m_1 = 2$. The subtraction of $((x_2 - x_3) + tm_3)$ instead of P has the advantage that t is available before P is available, saving one k-bit CPA delay.

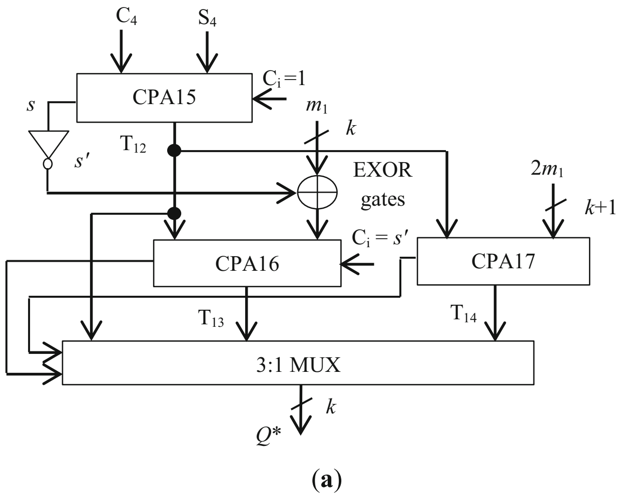
The computation of Eq. (8) requires addition of x_3, x_1, $(2x_2)_{2C}$ (realized as addition of one's complement of $2x_2$ and carry input of 1) and $t \times (2)_{2C}$ (two's complement of 2 enabled by t) using CSA3 and CSA4 followed by a modulo m_1 adder. The maximum positive and minimum negative values of $(x_3 - 2x_2 + x_1 - 2t)$ are $(4m - 4)$ and $(-4m + 2)$, respectively. The maximum positive value occurs when $x_1 = (2m - 2)$ and $x_3 = 2m$ and since $x_2 < x_3$ in this case, $t = 1$, thus making the maximum positive value $(4m - 4)$. On the other hand, when $x_1 = x_3 = 0$ and for all x_2 values, $t = 0$, yielding the minimum negative value $-2(2m - 1) = -4m + 2$. Hence, for modulo m_1 reduction of the sum of the outputs S_4 and C_4 of CSA4, at most addition of m_1 or $2m_1$ or subtraction of m_1 (addition of two's complement of m_1) is needed. This can be realized using a HS version of a parallel-type modulo m_1 adder that uses a 4:1 MUX to select the correct result Q^* as shown in figure 5. The CPA10 computes sum $T_9 = C_4 + S$ whereas the CSA5 followed by CPA11 computes $T_8 = C_4 + S - m_1$, CSA6 followed by CPA12 computes $T_{10} = C_4 + S + m_1$ and CSA7 followed by CPA13 computes $T_8 = C_4 + S + 2m_1$. Note that one's complement of m_1 is added with a carry input of $C_i = 1$ inserted in the free LSB of CARRY vector C_5. The correct result Q^* is selected using a 4:1 MUX as shown in BLOCK2 of figure 5. Next, the multiplication of Q^* with g $(= m)$ (see figure 5) is carried out to obtain the mixed radix digit Q using MOD-MUL block shown in figure 1d with $H = Q^*$. We next compute X as

$$X = x_2 + Pm_2 + Qm_2m_3 \tag{9}$$

using BLOCK3 (similar to BLOCK1 in figure 3). This uses multipliers MULT3 and MULT4 of sizes $k \times k$ and $2k \times k$, respectively, followed by CPA14. Note that MULT3 is a merged multiplier.

In the design of reverse converters following figure 4, we can notice that the computation of mixed radix digit Q is the critical path. Hence we present some alternate designs for computing the mixed radix digit Q employing two different methods for mod m_1 reduction of sum of C_4 and S_4 in figure 5 to obtain Q^*. In the design shown in figure 6a, we first add C_4 and S_4 in CPA15 to obtain T_{12}. Note that CPA15 has a carry input of 1 to realize two's complement of $2x_2$. We reduce the result T_{12} mod m_1 using one ADD/SUB unit realized by CPA16 and k exclusive-OR gates and one adder adding $2m_1$ using CPA17. The correct result is selected using a 3:1 MUX based on the sign bits of outputs of CPA16 and CPA17. Note that the exclusive-OR gates invert the bits of m_1 to facilitate subtraction and a carry input $C_i = s'$ is added where s is the sign bit of T_{12}. This block can be used in the architecture of converter D7 in figure 5 in place of BLOCK2 to realize converter D8.

In an alternative converter design D9, we use a binary-to-RNS converter to reduce T_{12} mod m_1 as shown in figure 6b. Since T_{12} is $(k + 2)$-bit wide, based on the two MSB bits, we add a constant W to the k-bit LSBs of T_{12}. Denoting $x = 2^k$ mod m_1 it can be seen that the two MSBs correspond to the four values before mod m_1 reduction: $00_b \rightarrow 0$, $01_b \rightarrow 2^k = x$, $10_b \rightarrow -2^{k+1}$, $11_b \rightarrow -2^k$. (Note that b indicates binary representation and $(k + 1)$th bit is sign bit of T_{12})). Thus, using a 4:1 MUX, appropriate value among these can be selected and added with k LSBs of T_{12} and



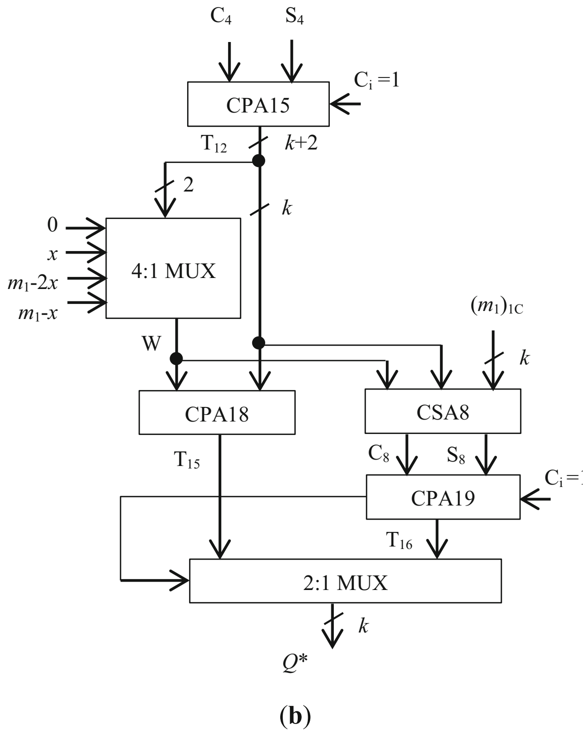


Figure 6. Alternative designs for replacing BLOCK2 in converter D7 for computation of Q^* (**a**) for converter D8 and (**b**) for converter D9.

reduced mod m_1 using CPA18, CSA8, CPA19 and 2:1 MUX to obtain Q^*. Note that the sum of W and word corresponding to k LSBs of T_{12} is at most $(2^k - 1)$ so that a single subtraction of modulus m_1 (addition of $(m_1)_{1C}$ with a carry input of 1 to CPA19) is sufficient to obtain Q^* as shown in figure 6b.

Next, we consider realizing the computation of Q^* and multiplication with m in a single block, instead of the cascade designs considered in figures 5 and 6. In the design D10, to determine the mixed radix digit Q, we need to compute $[m \times ((x_3 - 2x_2 + x_1 - 2t) \bmod m_1)] \bmod$

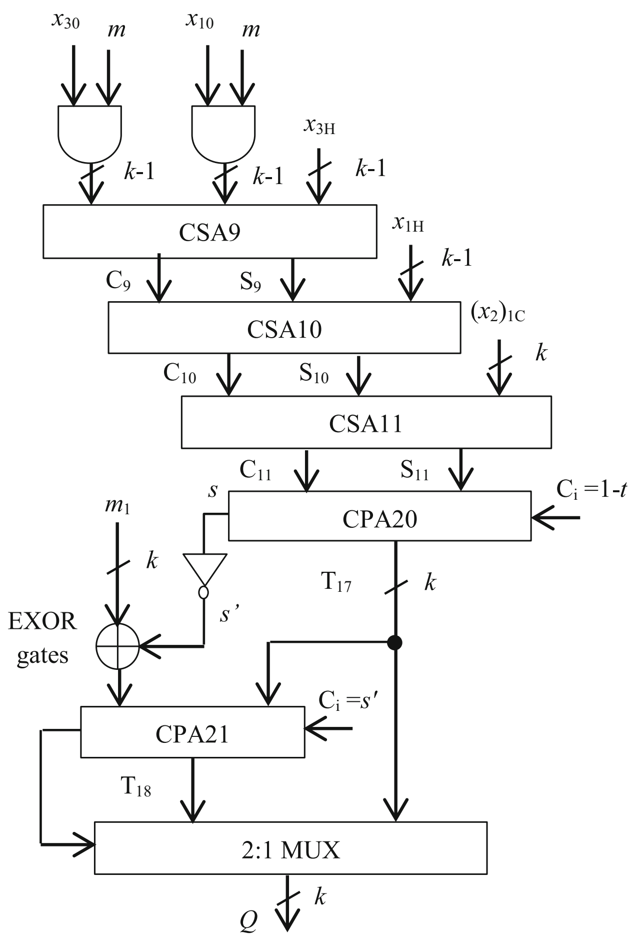


Figure 7. Architecture for the computation of mixed radix digit Q in D10.

$m_1 = (mx_3 + mx_1 - x_2 - t)$ mod m_1 in one step. Note that $(m \times 2x_2)$ and $(m \times 2t)$ are reduced modulo m_1 as x_2 and t, respectively, since $(2m)$ mod $x_1 = 1$. We consider $x_3 = 2x_{3H} + x_{30}$ and $x_1 = 2x_{1H} + x_{10}$ where x_{3H} and x_{1H} are the words formed by the most significant $(k - 1)$ bits of x_3 and x_1, respectively. The computation of $(mx_3 + mx_1)$ can be realized by adding $m \times x_{30}$, $m \times x_{10}$, x_{3H} and x_{1H} since $(2m \times x_{3H})$ mod $m_1 = x_{3H}$ and $(2m \times x_{1H})$ mod $m_1 = x_{1H}$. Note that $-x_2 - t$ is realized as $(x_2)_{1C} + (1 - t) = (x_2)_{1C} + t'$ where t' is inverted bit t. Thus, we need to compute $(m \times x_{30} + m{\times}x_{10} + x_{3H} + x_{1H} + x_{21C} + t')$ mod m_1 to obtain Q. Note that $m \times x_{30}$ and $m \times x_{10}$ can be obtained using a pair of $(k - 1)$ AND gates enabled by x_{30} and x_{10}, respectively, as shown in figure 7.

The five operands can be added using three-level CSA tree (CSA9–CSA11) and CPA20 followed by a mod m_1 adder. Note that the maximum positive and minimum negative values of the result of CPA20 are $(4m - 4)$ and $(-2m + 1)$, respectively. Hence, at most a single addition or subtraction of modulus m_1 is sufficient to obtain Q. Hence, a modulo m_1 adder using an ADD/SUB unit formed by CPA21, k exclusive-OR gates and a 2:1 MUX is used to compute Q.

4. Performance evaluation and comparison

The hardware requirement and conversion time for the various reverse converters described in [5, 18, 19, 21] for the moduli set M1 along with the proposed reverse converters have been presented in table 1. Note that FA, HA, AND and w:1 MUX stand for a full adder, half adder, two-input AND gate and w:1 multiplexer, respectively. The notations L1 and L2 are used to represent $2k \times k$ and $k \times k$ multipliers, respectively, and LiM (for i = 1, 2) is used to represent merged multiplier [23]. Note that the hardware requirement of L1 and L2 is $(2k^2 - 2k)$FA and $(k^2 - k)$FA, respectively, considering that an array multiplier using $(k - 2)$ carry save levels followed by a CPA is used and the delay of L1 and L2 is $(3k - 2)\Delta_{FA}$ and $(2k - 2)\Delta_{FA}$, respectively.

The converter D1 due to Premkumar [5] uses CRT. It needs five two-input adders, three $2k \times k$ multipliers each of the range $4m^2$ and 5 numbers of $3k$-bit 2:1 MUXs. Premkumar *et al* [18] suggested two converters D2 (Architecture A) and D3 (Architecture B) later by simplifying the conventional CRT. In this method, the modulo M reduction needed in [5] is simplified as modulo $(m_1 \times m_3)$ reduction. This converter needs one $2k \times k$ and another $k \times k$ multiplier of the range $4m^2$ and $2m$, respectively. Architecture A (D2) presented in [18] needs seven two-input adders and $6k$-bit 2:1 MUXs whereas another Architecture B (D3) presented in [18], which is a HS version, needs nine adders and $5k$-bit 2:1 MUXs. In the converter D4 for M1 proposed by Wang *et al* [19] based on new CRT II technique, we need one $2k \times k$ multiplier and one $k \times k$ multiplier, a few adders and a few comparators. The recent converter D5 for M1 due to Gbolagade *et al* [21] is based on the modification of CRT. It realizes modulo m_1 reduction using several MUXs and comparators and it needs one $2k \times k$ multiplier and one $k \times k$ multiplier. The hardware requirement and conversion time for these five converters D1–D5 are presented as first five entries in table 1. The proposed converters are presented as D6–D10 in table 1.

Among all the converters using two multipliers L1 and L2 for the moduli set M1, converter D5 needs the least area and D4 needs the highest area. However, it may be noted that the area of multipliers L1 and L2 has quadratic dependence on k and hence, for large k, the area of the multipliers dominates the total area. All the converters need similar conversion time except converters D2, D3, D8 and D9. The converters D2 and D3 need larger conversion time than converters D8 and D9. The m and n values needed for realizing DRs ranging from 8-bit to 64-bit for various moduli sets are presented in table 2. As an illustration, m = 21 for 16-bit DR of M1 implies use of the moduli set {41,42,43}. We have also considered the three reverse converters D11–D13 for the four-moduli set M2 $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ [24–26] and two reverse converters D14 and D15 for the four-moduli set M3 $\{2^n - 1, 2^n,$

Table 1. Comparison of hardware requirement and conversion time of various reverse converters for the three- and four-moduli sets M1–M3.

Design	Moduli set	Hardware requirement	Conversion time
D1 [5]	M1	$15k$ FA + $15k$ 2:1MUX + 3 L1	$(6k + k/2)\Delta_{FA}$
D2 [18]	M1	$11k$ FA + $6k$ 2:1MUX + L1 + L2	$9k\Delta_{FA} + \Delta_{L1}^{\dagger} + \Delta_{L2}^{\dagger}$
D3 [18]	M1	$13k$ FA + $5k$ 2:1MUX + L1 + L2	$9k\Delta_{FA} + \Delta_{L1}^{\dagger} + \Delta_{L2}^{\dagger}$
D4 [19]	M1	$(18k + 18)$ FA + $2k$ 3:1MUX + k AND + L1 + L2	$(6k + 7)\Delta_{FA} + \Delta_{3:1MUX} + \Delta_{L1}^{\dagger} + \Delta_{L2}^{\dagger}$
D5 [21]	M1	$(8k + 9)$ FA + $(k + 2)$ 5:1MUX + $(k + 2)$ NAND + $(k + 2)$ EXOR + L1 + L2M	$(6k + 2)\Delta_{FA} + \Delta_{5:1MUX} + \Delta_{EXOR} + \Delta_{NAND} + \Delta_{L1}^{\dagger}$
D6	M1	$(12k + 8)$FA + $2k$ 2:1MUX + k 3:1MUX + $(k - 1)$AND + L1 + L2M	$(6k + 4)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{3:1MUX} + \Delta_{AND} + \Delta_{L1}^{\dagger}$
D7	M1	$(15k + 18)$ FA + k 2:1MUX + k 4:1MUX + $(2k - 1)$ AND + L1 + L2M	$(6k + 5)\Delta_{FA} + \Delta_{4:1MUX} + 2\Delta_{AND} + \Delta_{L1}^{\dagger}$
D8	M1	$(11k + 10)$ FA + k 2:1MUX + k 3:1MUX + $(2k - 1)$ AND + k EXOR + L1 + L2M	$(7k + 6)\Delta_{FA} + \Delta_{3:1MUX} + 2\Delta_{AND} + \Delta_{EXOR} + \Delta_{L1}^{\dagger}$
D9	M1	$(12k + 9)$ FA + $2k$ 2:1MUX + k 4:1MUX + $(2k - 1)$ AND + L1 + L2M	$(7k + 6)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{4:1MUX} + 2\Delta_{AND} + \Delta_{L1}^{\dagger}$
D10	M1	$(10k + 4)$ FA + $2k$ 2:1MUX + $(2k - 2)$ AND + k EXOR + L1 + L2M	$(6k + 3)\Delta_{FA} + \Delta_{2:1MUX} + \Delta_{EXOR} + \Delta_{L1}^{\dagger}$
D11 [24]	M2	$(n^2/2 + 7n/2 + 7n + 4)$FA + HA + 2 2:1MUX	$(11n + \alpha+8)\Delta_{FA}$*
D12 [25]	M2	$(9n + 5+((n - 4)(n + 1)/2))$FA + $2n$ EXNOR + $2n$ OR	$[(23n + 12)/2]\Delta_{FA}$
D13 [26]	M2	$(10n + 6+(n - 4)(n + 1)/2)$FA + $(6n + 2)$ EXNOR + $(6n + 2)$ OR +$(n + 3)$ 2:1 MUX + $(2n + 1)$ 3:1 MUX	$[(15n + 22)/2]\Delta_{FA}$
D14 [25]	M3	$(n^2 + 12n + 12)$FA + $2n$ EXNOR +$2n$ OR + $(4n + 8)$ 2:1MUX	$(16n + 22)\Delta_{FA}$
D15 [27]	M3	$(2n^2 + 11n + 3)$FA	$(11.5n + 2\log_2 n + 2.5)\Delta_{FA}$

*α indicates number of levels in CSA tree of $n/2 + 1$ inputs.

†The notations Δ_{L1} and Δ_{L2} indicate delay of $(2k \times k)$ and $(k \times k)$ multipliers, respectively, in terms of FAs.

Table 2. Values of 'm' and 'n' to be considered for various DRs of M1–M3.

Moduli set		8-bit DR	16-bit DR	24-bit DR	32-bit DR	48-bit DR	64-bit DR
M1	m	4	21	129	813	32769	1321123
M2	n	2	4	6	8	12	16
M3	n	3	5	7	9	13	17

Table 3. Hardware requirement and delay estimation based on unit gate model for various three- and four-moduli set reverse converters.

Design	Unit gate area	Unit gate delay (Δ_g)
D5	$21k^2 + 50k + 93$	$27k + 15$
D6	$21k^2 + 76k + 55$	$27k + 20$
D7	$21k^2 + 98k + 125$	$27k + 24$
D8	$21k^2 + 69k + 69$	$31k + 29$
D9	$21k^2 + 80k + 62$	$31k + 30$
D10	$21k^2 + 59k + 26$	$27k + 14$
D11	$3.5n^2 + 73.5n + 37$	$44n + 4\alpha + 32$
D12	$3.5n^2 + 64.5n + 4$	$46n + 24$
D13	$3.5n^2 + 96.5n + 50$	$30n + 44$
D14	$7n^2 + 102n + 108$	$64n + 88$
D15	$14n^2 + 77n + 21$	$46n + 8\log_2 n + 10$

$2^n + 1, 2^{n+1} + 1\}$ [25, 27] for the purpose of comparison. Note that they use the efficient RNS to binary converters for the three-moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ [9–12] followed by a two-moduli MRC to include the fourth modulus. The hardware resource and conversion time requirements in terms of basic gates for the proposed reverse converters along with the converters for M1–M3 are also presented using unit-gate model [28] in table 3 for the general case and for the six standard DRs in table 4. Note that the equivalent number of gates for full adder, half adder, 2:1 MUX, EXOR/EXNOR, AND and OR gates is considered as 7, 3, 3, 2, 1 and 1 and the delays are considered as $4\Delta_g$, $2\Delta_g$, $2\Delta_g$, $2\Delta_g$, Δ_g and Δ_g, respectively, where Δ_g is unit-gate delay.

From table 4, it can be observed that for all the standard DRs, among the considered three- and four-moduli sets the design D12 is preferable regarding lower hardware resource requirement and the converter D13 needs least conversion time among all the converters. Among the converters D5–D10 for moduli set M1, the proposed converter D10 needs the lowest hardware resources for DR 8 and 16 bits whereas converter D5 is better for 24-, 32-, 48- and 64-bit DRs. Regarding conversion time, for all considered standard DRs, D5 and D10 are better than other converters D6–D9. It can also be observed that the proposed converters D6–D10 need less conversion time than converters D14 and D15 for all the considered standard DRs.

The proposed converters D6–D10 as well as design D5 [21] were implemented using Cadence (Version 14.20), Compiler: RC 14.25 and synthesized using the Cadence Encounter tool using 180-nm technology. The post place and route results of area, conversion time and power dissipation for all these designs for DRs of 8, 16, 24, 32, 48 and 64 bits are presented in table 5.

Regarding hardware requirements, for 8- and 64-bit DRs, the design D5 is superior to the converters D6–D10 and the design D6 outperforms converters D5 and D7–D10 for 16-, 24-, 32- and 48-bit DRs. For 16-bit D7 and D8 and for 32-bit and 48-bit DR, converter D8 require less hardware resources than D5. The converter D9 is preferable compared with D5 regarding area for 16-, 24-, 32- and 48-bit DRs.

Regarding conversion time, for 8, 32 and 64 bits, D5 performs better than D6–D10 and for 16-, 24- and 48-bit DRs, D6 outperforms converters D5 and D7–D10. The converter D7 also requires less conversion time than D5 for 16-bit DR.

Table 4. Area and delay comparison for 8-, 16-, 24-, 32-, 48- and 64-bit DR three- and four-moduli set reverse converters using unit gate model.

	8-bit DR		16-bit DR		24-bit DR		32-bit DR		48-bit DR		64-bit DR	
Design	Area	Delay	Area	Delay	Area	Delay	Area	Delay	Area	Delay	Area	Delay
D5	629	123	1149	177	2244	258	3184	312	7012	474	11357	609
D6	695	128	1267	182	2440	263	3432	317	7416	479	11891	614
D7	853	132	1469	186	2708	267	3744	321	7860	483	12445	618
D8	681	153	1239	215	2391	308	3369	370	7311	556	11751	711
D9	718	154	1298	216	2483	309	3483	371	7491	557	11986	712
D10	598	122	1136	176	2258	257	3216	311	7098	473	11488	608
D11	198	124	387	212	604	304	849	396	1423	580	2109	764
D12	147	116	318	208	517	300	744	392	1282	576	1932	760
D13	257	104	492	164	755	224	1046	284	1712	404	2490	524
D14	447	280	793	408	1165	536	1593	664	2617	920	3865	1176
D15	378	161	756	259	1246	355	1848	449	3388	638	5376	825

Table 5. ASIC implementation results of various reverse converters for the three-moduli set M1.

	Design								
	D5 [21]			D6			D7		
Dynamic range	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)
8-bit	3765	2609	677	4015	2761	824	4820	2913	1047
16-bit	12507	5828	4545	11476	4809	4228	11759	5104	4327
24-bit	23168	8091	11389	22400	7995	12177	24243	8716	11249
32-bit	31617	9382	17984	30560	9393	19158	32612	10411	18849
48-bit	61236	14225	34934	58119	13853	38309	61322	15512	36394
64-bit	90897	17638	62140	91270	17672	72497	94729	19723	65173
	D8			D9			D10		
Dynamic range	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)
8-bit	5066	3764	1193	4780	3378	1048	4690	3678	1058
16-bit	12012	6623	4741	12042	5853	4451	12943	6218	4222
24-bit	23255	10264	11863	22816	9894	12039	23777	10065	12160
32-bit	31378	12574	20142	30869	11716	19209	32506	12022	18623
48-bit	59419	18278	38528	58851	17638	37715	61412	17066	36995
64-bit	92374	23141	70215	91672	22508	68755	91918	21138	66426

Regarding power dissipation, 8-, 32-, 48- and 64-bit DRs, the converter D5 is superior than D6–D10 and for 16-bit DR, the converters D6, D7, D9 and D10 outperform converter D5. For 24-bit DR, the converter D7 needs less power dissipation than D5, D6 and D8–D10. Among the proposed converters, for 8-bit DR, the converter D6 is preferable and for 16- and 32-bit DRs, the converter D10 is superior to the other converters regarding power dissipation. For 24-, 48- and 64-bit DRs, the converter D7 needs least power dissipation compared with other proposed converters.

5. Conclusions

In this paper we have presented RNS-to-binary converters for the moduli set $\{2m-1, 2m, 2m+1\}$ using MRC technique. All the proposed converters were evaluated based on the hardware resource requirement as well as conversion time with all converters described in literature for the moduli set $\{2m-1, 2m, 2m+1\}$. The proposed converters have also been compared to two four-moduli reverse converters. All the proposed converters are implemented and compared to the area-efficient converter [21] for M1 regarding area and conversion time for different DRs. The proposed converters also need less conversion time than reverse converters for some four-moduli sets. The proposed converters for M1 were shown to be better than some of the other converters regarding area and conversion time while having the advantage of availability of mixed radix digits.

References

[1] Szabo N S and Tanaka R I 1967 *Residue Arithmetic and Its Applications to Computer Technology*. New York: Mc-Graw Hill

[2] Ananda Mohan P V 2016 *Residue Number Systems: Theory and Applications*. Basel: Birkhauser

[3] Omondi A and Premkumar A B 2007 *Residue Number System Theory and Implementation*, vol. 2. London: Imperial College Press

[4] Soderstrand M A, Jullien G A, Jenkins W K and Taylor F (Eds.) 1986 *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. Piscataway: IEEE Press

[5] Premkumar A B 1992 An RNS to binary converter in $\{2n+1, 2n, 2n-1\}$ moduli set. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 39: 480–482

[6] Premkumar A B 1995 An RNS to binary converter in a three moduli set with common factors. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 42: 298–301

[7] Andraros S and Ahmad H 1988 A new efficient memory-less residue to binary converter. *IEEE Trans. Circ. Syst.* 35: 1441–1444

[8] Piestrak S J 1995 A high-speed realization of residue to binary system conversion. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 42: 661–663

[9] Dhurkadas A 1998 Comments on 'A high-speed realization of a residue to binary number system converter'. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 45: 446–447

[10] Bhardwaj M, Premkumar A B and Srikanthan T 1998 Breaking the $2n$-bit carry propagation barrier in residue to binary conversion for the $\{2^n-1, 2^n, 2^n+1\}$ moduli set. *IEEE Trans. Circ. Syst. I: Fund. Theor. Appl.* 45: 998–1002

[11] Wang Z, Jullien G A and Miller W C 2000 An improved residue to binary converter. *IEEE Trans. Circ. Syst. I: Fund. Theor. Appl.* 47: 1437–1440

[12] Wang Y, Song X, Aboulhamid M and Shen H 2002 Adder based residue to binary number converters for $\{2^n - 1, 2^n, 2^n + 1\}$. *IEEE Trans. Signal Process.* 50: 1772–1779

[13] Hiasat A A and Abdel-Aty-Zohdy H S 1998 Residue to binary arithmetic converter for the moduli set $\{2^k, 2^k - 1, 2^{k-1} - 1\}$. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 45: 204–209

[14] Wang W, Swamy M N S, Ahmad M O and Wang Y 2000 A high-speed residue-to-binary converter for three moduli $\{2^k, 2^k - 1, 2^{k-1} - 1\}$ RNS and a scheme for its VLSI implementation. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 47: 1576–1581

[15] Wang W, Swamy M N S, Ahmad M O and Wang Y 2002 A note on 'A high-speed residue-to-binary converter for thee moduli $\{2^k, 2^k - 1, 2^{k-1} - 1\}$ RNS and a scheme for its VLSI implementation. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 49: 230

[16] Ananda Mohan P V 2008 New residue to binary converters for the moduli set $\{2^k, 2^k - 1, 2^{k-1} - 1\}$. In: *Proceedings of the IEEE Region 10 Conference (TENCON 2008)*, pp. 1–6

[17] Chaves R and Sousa L 2004 $\{2^n + 1, 2^{n+k}, 2^n - 1\}$: a new RNS moduli set extension. In: *Proceedings of the Euromicro Symposium on Digital System Design (DSD): Architectures, Methods and Tool*, pp. 210–217

[18] Premkumar A B, Bhardwaj M and Srikanthan T 1998 High-speed and low-cost reverse converters for the $\{2n - 1, 2n, 2n + 1\}$ moduli set. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 45: 903–908

[19] Wang Y, Swamy M N S and Ahmad M O 1999 Residue-to-binary number converters for three moduli sets. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 46: 180–183

[20] Gbolagade K A and Cotofana S D 2008 An efficient RNS to binary converter using the moduli set $\{2n - 1, 2n, 2n + 1\}$. In: *Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems (DCIS)*

[21] Gbolagade K A, Voicu G R and Cotofana S D 2011 An efficient FPGA design of residue-to-binary converter for the moduli set $\{2n + 1, 2n, 2n–1\}$. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 19: 1500–1503

[22] Wang Y 2000 Residue to binary converters based on New Chinese Remainder theorems. *IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process.* 47: 197–205

[23] Swartzlander Jr. E E 1980 Merged arithmetic. *IEEE Trans. Comput.* 29: 946–950

[24] Cao B, Srikanthan T and Chang C H 2005 Efficient reverse converters for the four-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1\}$. *IEE Proc. Comput. Digit. Tech.* 152: 687–696

[25] Ananda Mohan P V and Premkumar A B 2007 RNS to binary converters for two four moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$. *IEEE Trans. Circ. Syst. I: Reg. Papers* 54: 1245–1254

[26] Hosseinzadeh M, Molahosseini A and Navi K 2008 An improved reverse converter for the moduli set $\{2^n + 1, 2^n - 1, 2^n, 2^{n+1} - 1\}$. *IEICE Electron. Exp.* 5: 672–677

[27] Sousa L, Antao S and Chaves R 2013 On the design of RNS reverse converters for the four-moduli set $\{2^n + 1, 2^n - 1, 2^n, 2^{n+1} + 1\}$. *IEEE Trans. VLSI Syst.* 21: 1945–1949

[28] Bakalis D, Vergos H T and Spyrou A 2011 Efficient modulo $2n \pm 1$ squarers. *Integr. VLSI J.* 44: 163–174