New Reverse converters for the four-moduli set $\{2^n,2^n\text{-}1,2^n\text{+}1,2^{n\text{-}1}\text{-}1\}$ for n even

P. S. Phalguna
Dept. of Electronics and Communication Engg., Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal 576104, India
phalguna.ps@learner.manipal.edu

Dattaguru. V. Kamat
Dept. of Electronics and Communication Engg., Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal 576104, India
dv.kamath@manipal.edu

P. V. Ananda Mohan
Dept. of R&D, Centre for Development of Advanced Computing, Bengaluru 560 038, India
anandmohanpv@live.in

***Abstract*—In this paper, two reverse converters for the four-moduli set $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^{n\text{-}1}\text{-}1\}$ are described. One of these is based on Mixed Radix Conversion (MRC). Another converter is based on two-stage MRC in which two pairs of moduli are considered and intermediate results are obtained using MRC. A second stage uses MRC to obtain the final decoded number from these intermediate results. Both the converters are compared with previously reported converter for this moduli set regarding hardware resources and conversion time. Synthesis results on FPGA and ASIC are also presented.**

Keywords—RNS, Reverse converters, four-moduli sets, Mixed Radix Conversion

I. Introduction

In Residue number system (RNS), a moduli set is chosen such that the product of all these moduli is greater than the desired dynamic range [1]. The residues are remainders obtained by dividing a given binary number with the moduli. Several four-moduli sets have been described in literature using mutually prime moduli of one of the forms 2^a+1, 2^b-1 and 2^c with the aim of realizing higher dynamic range while retaining the word length between n to $2n$ bits. Several reverse converters [2]-[8] have been described for these RNS using four or more moduli which are based on horizontal and vertical extensions of a popular three-moduli set M0 $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ for realizing larger dynamic range than possible with the moduli set M0 [1]. In this paper, we consider the four-moduli set M1 $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^{n\text{-}1}\text{-}1\}$ for which two converters have been described using a front-end converter for moduli set M0 [5], [6] followed by MRC for the composite moduli set M0 and $2^{n\text{-}1}$-1. In [5], the authors also have suggested a two-stage converter but have not developed the converter fully. In this paper, we consider a two-stage converter and also a one-stage full-MRC based converter. We evaluate these two converters regarding the hardware resources and conversion time and compare them with previously reported converter for this moduli set. Synthesis results on ASIC and FPGA are also presented.

II. Background Material

The most popular techniques for Reverse conversion are based on Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC) [1]. In this paper, we use MRC for deriving reverse converters. The binary number X corresponding to given residues (x_1, x_2, x_3, x_4) for a four-moduli set $\{m_1, m_2, m_3, m_4\}$ can be obtained using MRC as

$$X = x_1 + U_A m_1 + U_D m_1 m_2 + U_F m_1 m_2 m_3 \quad (1)$$

where

$$U_A = \left|(x_2 - x_1)\left(\frac{1}{m_1}\right)_{m_2}\right|_{m_2},\ U_D = \left|\left((x_3 - x_1)\left(\frac{1}{m_1}\right)_{m_3} - U_A\right)\left(\frac{1}{m_2}\right)_{m_3}\right|_{m_3},$$

$$U_F = \left|\left(\left((x_4 - x_1)\left(\frac{1}{m_1}\right)_{m_4} - U_A\right)\left(\frac{1}{m_2}\right)_{m_4} - U_D\right)\left(\frac{1}{m_3}\right)_{m_4}\right|_{m_4} \quad (2)$$

This needs three steps each involving several modulo m_i subtractions and modulo multiplications with the multiplicative inverses. Note that $0 \le X < M$ in (1) where $M = m_1 m_2 m_3 m_4$.

In this paper, MRC technique is used. We need computation of (a-b) mod (2^k-1) where the words a and b are of length k-bits and l-bits respectively and $l \ge k$. In the case $l = k$, (a-b) mod (2^k-1) can be computed by adding one's-complement of b denoted as b_{1C} with a in an end-around-carry (EAC) adder as shown in Fig. 1(a). This needs only one k-bit carry-propagate-adder (CPA) needing k full-adders (FAs) with addition time $2kD_{FA}$ where D_{FA} is the delay of a full-adder. In case $l > k$, denoting B as $b_{l-1}b_{l-2}\ldots b_k b_{k-1}\ldots b_1 b_0$, we need to add with a, two k-bit words $B_{1(1C)}$ and $B_{2(1C)}$ where 1C means one's complement and $B_1 = b_{k-1}\ldots b_1 b_0$ and $B_2 = 0..0 b_{l-1} b_{l-2}\ldots b_{k+1} b_k$ in a k-bit carry-save-adder (CSA) followed by a k-bit CPA with EAC as shown in Fig.1(b).

III. Proposed Reverse converters using MRC

In this section, we present two reverse converters Converter 1 and Converter 2 for the four-moduli set M1 $\{2^n, 2^n\text{+}1, 2^n\text{-}1, 2^{n\text{-}1}\text{-}1\}$ for n even using conventional MRC and two-level MRC techniques respectively.

A. Converter-1

The MRC architecture for moduli set M1 is shown in Fig. 2. We denote the residues corresponding to the four-moduli $m_1 = 2^n$, $m_2 = 2^n+1$, $m_3 = 2^n-1$ and $m_4 = 2^{n-1}-1$ as x_1, x_2, x_3 and x_4 respectively. The dynamic range (DR) of M1 is $2^n(2^{2n}-1)(2^{n-1}-1)$. The various multiplicative inverses needed in (2) and Fig. 2 are as follows:

$$l_1 = \left(\frac{1}{2^n}\right)_{2^n+1} = (-1)_{2^n+1}, \qquad l_2 = \left(\frac{1}{2^n}\right)_{2^n-1} = 1,$$

$$l_3 = \left(\frac{1}{2^n}\right)_{2^{n-1}-1} = 2^{n-2}, \qquad l_4 = \left(\frac{1}{2^n+1}\right)_{2^n-1} = 2^{n-1},$$

$$l_5 = \left(\frac{1}{2^n+1}\right)_{2^{n-1}-1} = \frac{1}{3}(2^n-1),\ l_6 = \left(\frac{1}{2^n-1}\right)_{2^{n-1}-1} = 1. \quad (3)$$

The decoded number X_A for the four-moduli set $\{2^n, 2^n\text{+}1, 2^n\text{-}1, 2^{n\text{-}1}\text{-}1\}$ can be expressed following (1) as

$$X_A = x_1 + U_A 2^n + U_D 2^n(2^n+1) + U_F 2^n(2^n+1)(2^n-1) \quad (4)$$

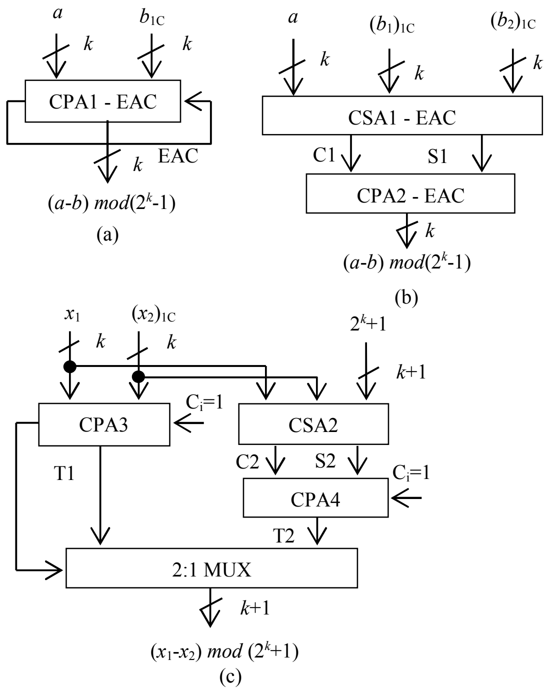

Fig.1.Modulo (2^k-1) subtractors(a) case $l = k$(MODSUBA), (b) case $l>k$(MODSUBB), (c) Modulo (2^k+1) subtractor (MODSUBC)

where U_A, U_D, U_F and x_1 are the mixed radix digits defined in (2).

Since $l_1 = -1$, we can compute $U_A = (x_1-x_2) \bmod (2^n+1)$ using a high-speed modulo (2^n+1) adder (MODSUBC) shown in Fig. 1(c) by considering x_1 as $(n+1)$-bit word by appending a most significant bit (MSB) of zero. Note that, in MODSUBC, $T1 = (x_1-x_2)$ and $T2 = ((x_1-x_2) + (2^n+1))$ are computed in parallel and based on the sign of T1 we select either T1 or T2 using a 2:1 MUX. The computation of $U_B = (x_3-x_1) \bmod (2^n-1)$ can be carried out by using MODSUBA shown in Fig. 1(a) to obtain U_B directly since the multiplicative inverse l_2 is 1. The computation of $U_C^* = (x_4-x_1) \bmod (2^{n-1}-1)$ is carried out by MODSUBB1 block of Fig. 1(b). The intermediate result U_C can be computed from U_C^* by performing circular-left-shift (CLS) by $(n-2)$-bits since l_3 is 2^{n-2}. Next, the intermediate result $U_D^* = (U_B-U_A) \bmod (2^n-1)$ can be computed using MODSUBB2 block of Fig. 1(b). The mixed radix digit U_D can be computed from U_D^* by performing CLS by $(n-1)$-bits since l_4 is 2^{n-1}. Note that $U_E^* = (U_C-U_A) \bmod (2^{n-1}-1)$ can be computed using modulo subtractor MODSUBB3 of Fig. 1(b).

Next, $U_E = (U_E^* \times l_5) \bmod (2^{n-1}-1)$ can be obtained using a MODMUL block as shown in Fig. 2. As an illustration, for n = 4, 6, 8, 10, we have l_5 = 5, 21, 85, 341 consisting of alternate 1's in the binary representation (from LSB to left) having 2, 3, 4 and 5 bits respectively which are '1'. Thus, the modulo $(2^{n-1}-1)$ addition of $n/2$ partial products (*PP*s) is needed to obtain the intermediate result U_E form U_E^*. Note that the partial products obtained extending beyond $(n-1)$-bits up to $(2n-3)$-bits need to be mapped in the least significant bit position due to the mod $(2^{n-1}-1)$ operation. Next, these words need to be added using a CSA tree with EAC to obtain U_E. In general, $(n/2)-2$ levels of

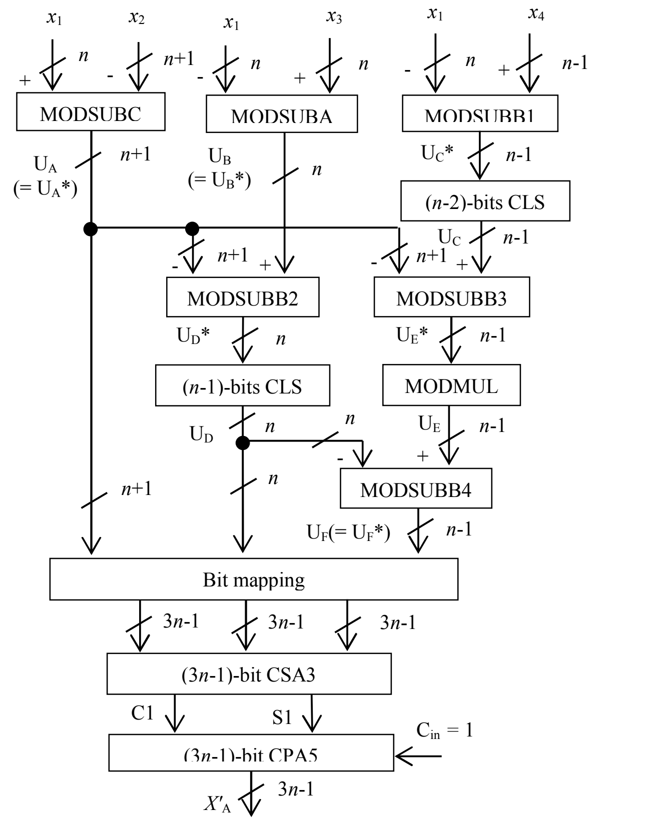

Fig. 2. Architecture of the converter 1 to obtain X'_A

CSA are needed. The computation of $U_F = (U_E-U_D) \bmod (2^{n-1}-1)$ is carried out using the modulo subtractor MODSUBB4 of Fig. 1(b). The mixed radix digit U_F is thus already available as U_F^* since l_6 is 1. The last stage of the converter computes $X'_A = (X_A-x_1)/m_1$, given as

$$X'_A = U_A + U_D\, 2^n + U_D + U_F 2^{2n} - U_F \tag{5}$$

Next, the decoded integer X_A can be directly obtained by appending x_1 to X'_A as least significant bits.

B. Converter-2

In this converter, in the first stage, we use MRC on the two moduli sets M_{14} $\{2^n, 2^{n-1}-1\}$ and M_{23} $\{2^n+1, 2^n-1\}$ to obtain the two intermediate numbers X_{14} and X_{23} respectively. Next, in second level, we use MRC for the residues (X_{14}, X_{23}) in the composite two moduli set $\{M_{14}, M_{23}\}$ where $M_{14} = m_1 \times m_4$ and $M_{23} = m_2 \times m_3$ to obtain X_A as shown in Fig. 3(a). Note that this approach was suggested in [5] but not investigated fully.

Computation of X_{14}:

The intermediate number X_{14} is computed as

$$X_{14} = x_1 + m_1 (a_1 b_1)_{m_4} = x_1 + 2^n (a_1 b_1)_{2^{n-1}-1} \tag{6}$$

where

$$a_1 = (x_4 - x_1)_{2^{n-1}-1},\ b_1 = \left(\frac{1}{2^n}\right)_{2^{n-1}-1} = 2^{n-2} \tag{7}$$

Since the word lengths of x_1 and x_4 are different, a_1 can be estimated using the modulo subtractor MODSUBB5 of Fig. 1(b). Next, $p_1 = (a_1 b_1) \bmod (2^{n-1}-1)$ where $b_1 = 2^{n-2}$ can be realized by CLS of a_1 by $(n-2)$-bits (see Fig. 3(b)). The result

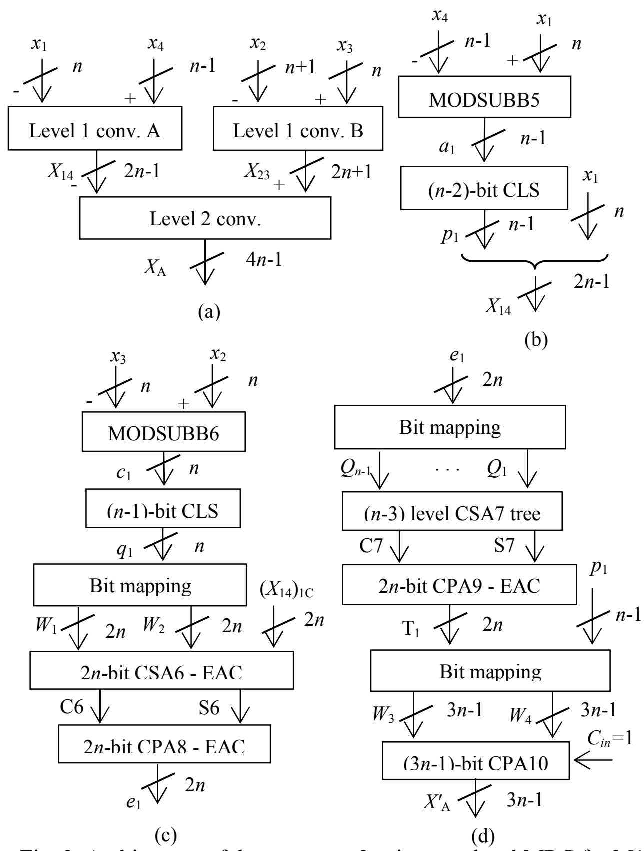

Fig. 3. Architecture of the converter 2 using two-level MRC for M1

p_1 can be concatenated with x_1 as n LSBs to yield a $(2n-1)$-bit word X_{14} following (6). The architecture for computing X_{14} is presented in Fig. 3(b).

Computation of X_{23}:

The intermediate number X_{23} is computed as

$$X_{23} = x_2 + (c_1 d_1)_{m_3} = x_2 + (2^n + 1)(c_1 d_1)_{2^n - 1} \quad (8)$$

where

$$c_1 = (x_3 - x_2)_{2^n - 1}, d_1 = \left(\frac{1}{2^n + 1}\right)_{2^n - 1} = 2^{n-1} \quad (9)$$

Note that c_1 can be obtained using the modulo subtractor (MODSUBB6) of Fig. 1(b). Next, $q_1 = (c_1 d_1)_{2^n - 1}$ can be realized by $(n-1)$-bit CLS of c_1. Next, the computation of X_{23} needs mod $(2^{2n}-1)$ addition of W_1 and W_2 where W_1 $(= q_1 2^n + q_1)$ and W_2 $(= x_2)$ are the two $2n$-bit words: $W_1 = q_{1,n-1} \ldots q_{1,1} q_{1,0} q_{1,n-1} \ldots q_{1,1} q_{1,0}$ and $W_2 = 0 \ldots 0 x_{2,n} \ldots x_{2,2} x_{2,1} x_{2,0}$.

Computation of X_A:

The decoded integer X_A is computed as

$$X_A = X_{14} + M_{14}(e_1 f_1)_{M_{23}} \quad (10)$$

where

$$e_1 = (X_{23} - X_{14})_{M_{23}}, f_1 = \left(\frac{1}{M_{14}}\right)_{M_{23}} = \frac{1}{3}(2^{2n+1} - 2^{n+2} - 2^2) = \sum_{i=0}^{\left(\frac{n}{2}-2\right)} 2^{2(n-i)-1} + \sum_{i=1}^{\frac{n}{2}} 2^{2i} = 2^n + \{(2^n + 1)Z\} \quad (11)$$

Where $Z = (2^{n-2} + 2^{n-4} + \ldots + 2^2)$. The multiplicative inverse f_1 can be verified to be true since $(M_{14}) \times f_1 = 1$ mod M_{23}. As an illustration, for n = 4, 6, 8, 10 we have f_1 = 148, 2644, 43348, 697684 respectively, having $(n-1)$-bits which are ‘1’.

Table I. Hardware and time evaluation of various components involved in converters D1 and D2

Design	Component	Hardware requirement	Time
D1	MODSUBA	nA_{FA}	$2nD_{FA}$
	MODSUBB(1&4)	$nA_{FA}+A_{HA}+(n-2)A_{EXNOR/OR}$	$(2n-3)D_{FA}+2D_{HA}$
	MODSUBB2	$nA_{FA}+A_{HA}+(n-1)A_{EXNOR/OR}$	$(2n-1)D_{FA}+2D_{HA}$
	MODSUBB3	$nA_{FA}+A_{HA}+(n-3)A_{EXNOR/OR}$	$(2n-3)D_{FA}+2D_{HA}$
	MODSUBC	$3nA_{FA}+3A_{HA}+(n+1)A_{2:1\ MUX}$	$(n+1)D_{FA}+D_{HA}+D_{2:1MUX}$
	MODMUL	$[((n-1)(n/2-2))+(n-1)]A_{FA}$	$[(2n-2)+(n/2-2)]D_{FA}$
	CSA3 & CPA5	$(4n-2)A_{FA}+2A_{EXNOR/OR}$	$3nD_{FA}$
D2	MODSUBB5	$nA_{FA}+(n-2)A_{EXNOR/OR}$	$(2n-1)D_{FA}$
	MODSUBB6	$(n+1)A_{FA}+(n-1)A_{EXNOR/OR}$	$(2n+1)D_{FA}$
	CSA6	$(2n-1)A_{FA}+A_{EXNOR/OR}$	D_{FA}
	CPA(8&9)	$(2n-1)A_{FA}+A_{HA}$	$(4n-2)D_{FA}+2D_{HA}$
	CSA7	$(2n^2-6n)A_{FA}$	$(n-3)D_{FA}$
	CPA10	$2nA_{FA}+(n-1)A_{EXNOR/OR}$	$2nD_{FA}+(n-1)D_{EXNOR/OR}$

Next, the computation of e_1 can be carried out by adding one’s complement of X_{14} prepended with one zero bit, with W_1 and W_2 using a $2n$-bit CSA followed by CPA both with EAC as shown in Fig. 3(c). Note that computation of $T_1 = (e_1 f_1)_{M_{23}}$ needs addition of $(n-1)$ partial products (PPs) using $2n$-bit CSA tree having $(n-3)$ levels followed by a $2n$-bit CPA, both with EAC. Next, from (10), since the n LSBs of the decoded integer are given by x_1, we compute only X'_A given as

$$X'_A = \frac{X_A - x_1}{2^n} = p_1 + T_1(2^{n-1} - 1) \quad (12)$$

Thus in the final stage the three operands p_1, $(T_1 \times 2^{n-1})$ and one’s complement of T_1 can be simplified as two $(3n-1)$-bit words W_3 and W_4 which can be added using a $(3n-1)$-bit CPA along with a carry input of 1 to realize two’s complement of T_1. The architecture for computing intermediate result T_1 and X'_A is shown in Fig. 3(d). Next the decoded integer X_A can be directly obtained by concatenating X'_A with x_1 as LSBs.

IV. Performance Evaluation and Comparison

The hardware resource requirement and computation time of various components in the architectures of Fig. 2 and 3 are presented in Table I. The hardware requirements and conversion time in terms of basic gates as well as unit gate model for the converters D1 and D2 as well as Cao *et al.* 3-stage converter [5] D3 are presented in Table II. Note that Cao *et al.* design has used converter in [10] for the moduli set M0 and has used MRC for the composite moduli set. From Table II, we can notice that, the converter D1 needs less hardware resources than the converters D2 and D3. However, converter D3 needs less conversion time than converters D1 and D2. The two-stage converter D2 needs more hardware and conversion time than the converter D1. Note that in case of converter D2, $e_1 = [2^n + \{(2^n+1)Z\}]$ (see (11)) can be computed as addition of the three words: Z_a, $(Z_a \times 2^n)$ and $(e_1 \times 2^n)$ using a CSA followed by CPA with EAC. Since $Z_a = (Z \times e_1)$ has $(n/2-1)$ partial products only, these can be summed using a $(n/2-3)$ level CSA followed by a CPA both with EAC. Thus, $(e_1 \times f_1)$ mod M_{23} computation needs less hardware. Thus the constant multiplication (CM) [6] can be efficiently implemented.

The proposed converters D1, D2 as well as D3 [5] were implemented on Xilinx Virtex 6 xc6vhx380t-3-ff1923 FPGA using Xilinx ISE 14.5 tool. The number of slices and the conversion time (obtained by post-place and route static

Table II. Hardware resource requirements and conversion times of various converters

Design	Conversion method	Hardware requirement	Unit Gates requirement	Conversion time	Conversion Time (unit gate delays)
D1	Conventional MRC	$(n^2/2-n/2+11n-3)\ A_{FA} + 7A_{HA} + (4n-6)\ A_{EXNOR/OR} + (n+1)\ A_{2:1MUX}$	$(7/2)n^2 +88.5n-36$	$(n/2 + 11n - 15)\ D_{FA} + 6\ D_{HA}$	$46n-48$
D2	2-stage MRC	$(2n^2+4n-2)\ A_{FA} + 2\ A_{HA} + (3n-3)\ A_{EXNOR/OR}$	$14n^2+37n-15$	$(13n-5)\ D_{FA} + 4D_{HA} + (n-1)\ D_{EXNOR/OR}$	$54n-14$
D3 [5]	3-stage CE MRC(3-1)	$(n^2+7n-2)A_{FA}+A_{HA}+2A_{2:1MUX} +(2n-8)A_{XNOR}+6A_{AND}+(2n-8)A_{OR}+6A_{XOR}$	$7n^2+55n-11$	$(9n+m+1)$† D_{FA}	$36n+4m+4$

† Note that m the number of levels in a n-input CSA.

Table III. FPGA implementation results of converters D1, D2 and D3 for M1

Design	Dynamic Range											
	8-bit DR		16-bit DR		24-bit DR		32-bit DR		48-bit DR		64-bit DR	
	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)	Area (Slices)	Delay (ns)
D1	48	15.475	61	19.771	104	24.532	86	17.677	119	22.952	173	28.496
D2	31	14.800	47	13.712	76	20.846	80	15.926	116	15.433	163	17.115
D3 [5]	24	15.133	38	13.814	49	17.643	60	15.436	97	22.378	130	27.141

Table IV. ASIC implementation results of converters D1, D2 and D3 for M1

Dynamic Range	D1			D2			D3 [5]		
	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)	Area (μm^2)	Delay (ps)	Power (μW)
8-bit	4205	4524	1065	4178	4840	939	3054	5365	655
16-bit	7448	7055	2461	8143	8450	2703	5512	7885	1502
24-bit	10625	9444	4096	11420	13810	4975	7860	10394	2396
32-bit	14530	11783	6507	15518	17367	8656	11140	12760	3819
48-bit	22104	18652	12675	25716	21458	18082	17088	19771	8509
64-bit	30593	24013	19599	37582	29498	32094	24210	25354	14073

timing analysis) are provided in Table III for few DRs. From Table III, it can be observed that the conversion time of the converter D2 is less than that of converter D3 for 48-bit and 64-bit dynamic ranges by 31% and 36.94% respectively. For DRs of 8-, 16-, 32-bits both D2 and D3 exhibit similar conversion time. For 24-bit DR, D3 needs lowest conversion time. The converter D3 needs less hardware resources than converters D1 and D2 for all dynamic ranges. It may also be noted that converter D2 needs less hardware resources than those of converter D1 for all DRs.

The proposed converters D1, D2 as well as D3 [5] have also been implemented using Cadence (Version 14.20) Compiler: RC14.25 and synthesized using Cadence Encounter tool using 180 nm technology. The post-place and route results of area, conversion time and power dissipation for all the three converters are presented in Table IV. From Table IV, it can be seen that D1 needs conversion time less than that of converter D3 ranging between 5.6% to 15%. The converter D3 needs least hardware among all the converters. The converter D1 outperforms D2 and D3 regarding conversion time and D3 is preferable compared to D1 and D2 regarding power dissipation. The converter D2 needs more hardware resources, and power dissipation than converter D1 for all DRs except for DR of 8-bits. For all dynamic ranges, converter D2 needs more conversion time than converter D1.

V. Conclusions

In this paper, we have presented two Reverse converters for the four-moduli set $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^{n-1}\text{-}1\}$ (n even) using conventional MRC and two-stage MRC technique. Both the proposed converters were compared with Cao *et al*. converter. It has been shown that the proposed converters are better than Cao *et al*. converter in some cases regarding hardware requirements/conversion time/ power dissipation. It may be mentioned that vertical extension of moduli set M1 has been considered in [6] for which $k = n$ leads to Cao *et al*. design.

References

[1] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and implementation*, London: Imperial college Press, 2007.

[2] A.P. Vinod and A.B. Premkumar, "A residue to Binary converter for the 4-moduli superset $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n+1}\text{-}1\}$," *J. Circuits Syst. Comput*, vol. 10, no. 1&2, pp. 85-99, Feb. 2000.

[3] M. Bhardwaj, T. Srikanthan and C.T. Clarke, "A reverse converter for the 4-moduli super set $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n+1}\text{+}1\}$," in *Proc. 14th IEEE Symp.Computer Arithmetic*, Adelaide, Australia, Apr. 1999, pp. 168-175.

[4] P.V. A.Mohan and A.B. Premkumar, "RNS to Binary converters for two four moduli sets $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n+1}\text{-}1\}$ and $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n+1}\text{+}1\}$," *IEEE Trans. Circuits syst. I*, vol. 54, no. 6, pp. 1245-1254, Jun. 2007.

[5] B. Cao, T. Srikanthan, and C. H. Chang, "Efficient reverse converters for the four-moduli sets $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n+1}\text{-}1\}$ and $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{n-1}\text{-}1\}$," in *IEE Proc. Comput.& Digital Techniques*, vol. 152, no.5, pp. 687-696, Sep. 2005.

[6] P. Patronik and S.J. Piestrak, "Design of Reverse Converters for General RNS moduli sets $\{2^k, 2^n\text{-}1, 2^n\text{+}1, 2^{n+1}\text{-}1\}$ and $\{2^k, 2^n\text{-}1, 2^n\text{+}1, 2^{n-1}\text{-}1\}$ (n even)," *IEEE Trans. Circuits syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1687-1700, Jun. 2014.

[7] L. Sousa, S. Antao and R. Chaves, "On the design of RNS reverse converters for the four-moduli set $\{2^n\text{+}1, 2^n\text{-}1, 2^n, 2^{n+1}\text{+}1\}$," *IEEE Trans. Very Large Scale Integr.Systs.*,vol.21, no. 10. pp. 1945-1949, Oct. 2013.

[8] P. Patronik and S.J. Piestrak, "Design of Reverse Converters for the new RNS moduli set $\{2^n\text{+}1, 2^n, 2^n\text{-}1, 2^{n-1}\text{+}1\}$ (n odd)," *IEEE Trans. Circuits syst. I, Reg. Papers*, vol.61, no. 12, pp. 3436-3449, Dec. 2014.

[9] M. Esmaeildoust, K. Navi, M. Taheri, A.S. Molahosseini and S. Khodambashi, "Efficient RNS to Binary Converters for the new 4-moduli set $\{2^n, 2^{n+1}\text{-}1, 2^n\text{-}1, 2^{n-1}\text{-}1\}$", *IEICE Electronics Express*, vol. 9, no.1, pp. 1-7, 2012.

[10] Y. Wang, X. Song, M. Aboulhamid and H. Shen, "Adder based residue to binary number converters for $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$," *IEEE Trans.Signal Process.*, vol. 50, no. 7, pp. 1772-1779, Jul. 2002.