Design of an RNS Reverse Converter for a New Five-Moduli Special Set [*]

Piotr Patronik[1], Krzysztof Berezowski[1], Janusz Biernat[1], Stanisław J. Piestrak[2], Aviral Shrivastava[3]

[1]Inst. of Computer Engineering, Control, and Robotics, Wrocław Univ. of Technology, 50-372 Wrocław, Poland

[2]Lab. d'Instrumentation Électron. de Nancy (LIEN), Université de Lorraine, 54506 Vandœuvre-lès-Nancy, France

[3]Compiler Microarchitecture Lab, CSE Dept, Arizona State University, Tempe, AZ 85281, USA

{firstname.lastname}@pwr.wroc.pl, piestrak@univ-metz.fr, aviral.shrivastava@asu.edu

ABSTRACT

In this paper, we present a new residue number system (RNS) $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1, 2^{n-1}+1\}$ of five well-balanced moduli that are co-prime for odd n. This new RNS complements the 5-moduli RNS system proposed before for even n $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1, 2^{n-1}-1\}$. With the new set, we also present a novel approach to designing multi-moduli reverse converters that focuses strongly on moving a significant amount of computations off the critical path. The synthesis of the resulting design over the ST Microelectronics 65nm LP library demonstrates that the delay, area, and power characteristics improve the performance and power consumption of the existing complementary 5-moduli set.

Categories and Subject Descriptors

B.2.4 [**Arithmetic and Logic Structures**]: High-Speed Arithmetic — cost/performance

General Terms

Performance, Design, Theory

Keywords

Reverse Converter, 5-moduli Set, Residue Number System

1. INTRODUCTION

The residue number system (RNS) is a non-positional number system capable of improving performance and reducing power consumption of add/multiply-intensive integer datapaths by decomposing computations into r small magnitude, independent, and parallel residue channels. This results in reduced carry propagation in additions as well as size and depth of the partial product matrices in multiplications as compared to the classic 2's complement system (TCS). Consequently, the critical path delay, area, and power consumption may simultaneously reduce, leading to faster and more efficient circuits. RNS arithmetic is especially effective in digital signal and image processing applications [10, 11].

[*]This research was supported by Polish *Narodowe Centrum Nauki, NCN.* Grant no. DEC-2011/01/B/ST7/06120.
The research of S. J. Piestrak was done when he was with CAIRN Res. Team, IRISA/INRIA, 22305 Lannion, France, on leave from Univ. of Metz, 57070 Metz, France.

RNS datapath interacts with its TCS environment through forward and reverse conversions. Both require extra hardware, power, and time to be carried out. Especially the reverse conversion complexity grows significantly with increasing number of moduli, therefore, new RNSs are proposed along with their respective reverse converters, and each application requires careful balancing of their trade-offs [7].

Special moduli, even of the form 2^n and odd of the form 2^n-1 and 2^n+1, simplify the design of all basic arithmetic operations as well as forward and reverse conversions [7]. Therefore, many special moduli sets have been proposed. These include the classic three-moduli set $\{2^n+1, 2^n, 2^n-1\}$ [4,9] as well as four-moduli sets, e.g. $\{2^n+1, 2^n-1, 2^{n+1}\pm 1\}$ [1,2]. However, many of those already proposed, including five-moduli sets $\{2^n, 2^{2n+1}-1, 2^{n/2}-1, 2^{n/2}+1, 2^n+1\}$ [5] and $\{2^n, 2^{n/2}-1, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$ [6], suffer from significant imbalance in channels' magnitude, making one channel a computational bottleneck, while leaving other with an unutilized timing slack, consequently diminishing the very advantages of the modular decomposition.

While the 5-moduli set $\{2^n+1, 2^n, 2^n-1, 2^{n+1}-1, 2^{n-1}-1\}$ [3] addresses this problem, it is co-prime only for even n, and as such offers only a coarse coverage of its $(5n-1)$ bits of the dynamic range, stepping by 10 bits for subsequent even n. In this work, we propose a new special 5-moduli set $\{2^n+1, 2^n, 2^n-1, 2^{n+1}+1, 2^{n-1}+1\}$ that is co-prime for odd n and as such it is complementary to the one of [3]. More importantly, we propose a novel approach to designing multi-moduli reverse converters that: (i) attempts at moving non-critical computations off from the critical path, (ii) carries the constant modulo 2^n+1 multiplications in n-bit wide carry-save adder (CSA) trees, and (iii) employs a novel technique of subtraction-free computation of the final value.

2. RNS AND CONVERSION BASICS

An RNS is defined by a set of r pairwise co-prime integer *moduli* $\{m_1, m_2, \ldots, m_r\}$. Its *dynamic range* M equals to $M = \prod_{i=1}^{r} m_i$. An integer value $X \in [0, M-1]$ is uniquely represented by the r-tuple $\{x_1, \ldots, x_r\}$ of residues $x_i : x_i = \frac{X}{m_i}$ calculated with respect to $\{m_1, m_2, \ldots, m_r\}$.

Given X, Y represented by $\{x_1, \ldots, x_r\}$, $\{y_1, \ldots, y_r\}$, respectively, $Z = |X \circ Y|_M$ can be computed as $z_i = |x_i \circ y_i|_{m_i}$, $1 \le i \le r$ for $\circ \in \{+, -, \times\}$. Each RNS digit z_i of the result depends solely on x_i and y_i and its calculation does not involve other digits z_k for any $k \ne i$. That very parallelism is the primary advantage of applying the RNS.

There are two ways of calculating $|X|_M$ from $\{x_1, \ldots, x_r\}$ given $\{m_1, m_2, \ldots, m_r\}$. The first method is the *Chinese*

Reminder Theorem (CRT) typically formulated as

$$X = \left| \sum_{i=1}^{r} \hat{m}_i \left| x_i \hat{m}_i^{-1} \right|_{m_i} \right|_M \quad (1)$$

where $\hat{m}_i = \frac{M}{m_i}$ and $\hat{m}_i^{-1}$ is the *multiplicative inverse* of $\hat{m}_i$ [9], i.e., $|\hat{m}_i \cdot \hat{m}_i^{-1}|_{m_i} = 1$. The CRT is inherently parallel at the expense of a large multi-operand addition mod $\Pi_{i=1}^{r} m_i$, whose hardware implementation could be disadvantageous.

The second method is the *Mixed-Radix Conversion* (MRC)

$$X = x_1 + d_1 m_1 + d_2 m_1 m_2 + \ldots + d_{r-1}(m_1 m_2 \cdots m_{r-1}). \quad (2)$$

MRC calculates the mixed-radix digits $\{x_1, d_1, d_2, \ldots, d_{r-1}\}$ from residues and represents X over the mixed-radix base $\{1, m_1, m_1 m_2, \ldots, m_1 m_2 \cdots m_{r-1}\}$. Being inherently sequential, MRC has the advantage of computing modulo operations of a magnitude of single moduli throughout the whole conversion process. A special two-moduli case of (2) (2-MRC) is often used for constructing multi-moduli reverse converters in a recursive, modulus-by-modulus fashion

$$X = x_1 + m_1 \underbrace{\left|(x_2 - x_1) m_1^{-1}\right|_{m_2}}_{d_1}. \quad (3)$$

3. DESIGN OF THE CONVERTER

In our five-moduli RNS, we denote $m_1 = 2^n - 1$, $m_2 = 2^n$, $m_3 = 2^n + 1$, $m_4 = 2^{n+1} + 1$, and $m_5 = 2^{n-1} + 1$, while the residues corresponding to these moduli as x_1 through x_5. Similarly to [3] and many other, our converter carries out the conversion in the following three steps.

1) Convert the residues $\{x_1, x_2, x_3\}$ using an efficient CRT-based three moduli converter of [4]

$$X_1 = x_2 + 2^n \cdot \underbrace{\left|2^n(x_3 - x_2) + 2^{n-1}(2^n + 1)(x_1 - x_3)\right|_{2^{2n}-1}}_{X_h}$$

Note that $0 \leq X_1 < M_1$ and $M_1 = 2^n(2^{2n} - 1)$.

2) Apply 2-MRC to the pair of values $\{X_1, x_4\}$ over the two-moduli set $\{M_1, m_4\}$ to obtain a residue X_2, $0 \leq X_2 < M_2$, and the new module $M_2 = M_1 m_4$

$$X_2 = X_1 + M_1 \underbrace{|\overbrace{(x_4 - X_1)}^{L_4} M_1^{-1}|_{m_4}}_{R_4}. \quad (4)$$

3) Apply 2-MRC to the pair of values $\{X_2, x_5\}$ over the two-moduli set $\{M_2, m_5\}$ to obtain the final conversion result X, $0 \leq X < M$, where $M = M_2 m_5$

$$X = X_2 + M_2 \underbrace{|\overbrace{(x_5 - X_2)}^{L_5} M_2^{-1}|_{m_4}}_{R_5} \quad (5)$$

In the reminder of this paper, we develop the techniques to improve the efficiency of such conversion process.

3.1 Preliminaries

We leave without a proof the co-primality of the set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1, 2^{n-1} + 1\}$, as well as we assume the existence of X_1, i.e., the result of the conversion of residues $\{x_1, x_2, x_3\}$ with respect to moduli set $\{m_1, m_2, m_3\}$.

The inverses M_1^{-1} and M_2^{-1} have to satisfy respectively

$$\left|M_1^{-1} m_1 m_2 m_3\right|_{m_4} = 1 \quad (6)$$

$$\left|M_2^{-1} m_1 m_2 m_3 m_4\right|_{m_5} = 1 \quad (7)$$

In [1], it was observed that for $m_1 = 2^n - 1$, $m_2 = 2^n$, $m_3 = 2^n + 1$, and $m_4 = 2^{n+1} + 1$

$$M_1^{-1} = 2^n + \sum_{k=1}^{\frac{n-5}{2}} 2^{n-2k} + 2^4 - 2. \quad (8)$$

Similarly, if $m_5 = 2^{n-1} + 1$ then

$$M_2^{-1} = \left|18^{-1}\right|_{m_5}. \quad (9)$$

satisfies (7) for odd n. The bit-level structure of this inverse as a function of n is given in Figure 1.

3.2 Multiplication by a Constant mod $2^\alpha + 1$

Because both 2-MRC steps require multiplication by a constant, we take the uniform approach to this computation. Let an integer $X^* = (x_\alpha, \ldots, x_0)$ and a constant $C = (c_\alpha, \ldots, c_0)$ be $(\alpha + 1)$-bit wide mod $2^\alpha + 1$. Let $X'' = (x_{\alpha-1}, \ldots, x_0)$ be a vector of α least significant bits of X^*. Then we have

$$|C \times X^*|_{2^\alpha+1} = \left|C \sum_{i=0}^{\alpha} 2^i x_i\right|_{2^\alpha+1} = \left|C 2^\alpha x_\alpha + C \sum_{i=0}^{\alpha-1} 2^i x_i\right|_{2^\alpha+1}$$

Because $|2^\alpha|_{2^\alpha+1} = -1$, therefore

$$|C \times X|_{2^\alpha+1} = \left|\sum_{i=0}^{\alpha} c_i \underbrace{2^i \sum_{j=0}^{\alpha-1} 2^j x_j}_{2^i X''} - C x_\alpha\right|_{2^\alpha+1}$$

The value of $2^i X''$ can be easily computed by means of left circular shift with bit inversion and adding correction

$$2^i X'' = 2^i \sum_{j=0}^{\alpha-1} 2^j x_j = CLN_\alpha(X'', i) - (2^{i+1} - 1),$$

where $CLN_\alpha(U, p) = \left(u_{(\alpha-p)}, \ldots, u_0, \overline{u}_{(\alpha-1)}, \ldots, \overline{u}_{(\alpha-p+1)}\right)$ for any α-bit wide integer $U = (u_{\alpha-1}, \ldots, u_0)$ and $0 \leq p < \alpha$.

Thus, the multiplication $|C \times X^*|_{2^\alpha+1}$ can be computed as

$$\left|\sum_{i=0}^{\alpha} c_i CLN_\alpha(X'', i) - c_i \sum_{i=0}^{\alpha} (2^{i+1} - 1) - C x_\alpha\right|_{2^\alpha+1}$$

and implemented by a multioperand modular adder (MOMA) [8] of α least significant bits of X^* rotated with inversion, and a conditional correction depending on the value of the most significant bit x_α of X^*.

3.3 Calculation of 2-MRC

Subsequent applications of the 2-MRC produce deep computational paths that contribute to either delay or latency. We alleviate this disadvantage by decomposing each 2-MRC step into a critical and a non-critical component, and then move the latter off the critical path and compute it in parallel.

1) Calculation of L_4: $L_4 = |x_4 - X_1|_{m_4}$ from Equation (4) is computed from the 3-moduli converter output X_1 as

$$L_4 = |x_4 - X_1|_{2^{n+1}+1} = |x_4 - (2^n X_h + x_2)|_{2^{n+1}+1}$$
$$= |\underbrace{x_4 - x_2}_{L_4'} - 2^n X_h|_{2^{n+1}+1} \quad (11)$$

Because X_h is a $2n$-bit wide integer mod $2^{2n} - 1$, x_4 is an $(n + 2)$-bit wide integer mod $2^{n+1} + 1$, and x_2 is an n-bit wide integer mod 2^n, a direct computation of L_4 in an $(n + 1)$-bit wide CSA mod $2^{n+1} + 1$ requires six $(n + 1)$-bit wide operands. Observe that the inputs x_2 and x_4 are

$$\left|18^{-1}\right|_{2^{n-1}+1} = \begin{cases} 2^{n-2}+2^{n-3}-\sum_{i=0}^{k-2}2^{6i+7}+\sum_{i=0}^{k-2}2^{6i+4}-2^0 & \text{for } n=6k+1,\ k\geq 1 \\ \sum_{i=0}^{k}2^{6i}-\sum_{i=0}^{k-1}2^{6i+3} & \text{for } n=6k+5,\ k\geq 1 \\ 2^{n-4}+2^{n-3}+\sum_{i=0}^{k-1}2^{6i+2}-\sum_{i=0}^{k-2}2^{6i+5} & \text{for } n=6k+3,\ k\geq 1 \end{cases} \qquad (10)$$

Figure 1: The distribution of significant bits in $\left|18^{-1}\right|_{2^{n-1}+1}$

available much earlier than the output X_h of the 3-moduli converter. Therefore, the residues x_2, x_4 as well as the corrective constant (available at the design time) can be added separately and scheduled in parallel to the computation of X_h, thus forming a new component L'_4, so that each of additions needed to obtain L'_4 and X_h requires three operands only.

Let $x_{4,i}$ and $x_{2,i}$ denote the i-th bit of vectors x_4, and x_2, respectively. Then L'_4 can be computed as

$$\left| \begin{array}{l} (x_{4,n} \vee x_{4,(n+1)})x_{4,(n-1)}\dots x_{4,0} \ + \\ x_{4,(n+1)}\overline{x}_{2,(n-1)}\dots\overline{x}_{2,0} - 2^{n-2}+1 \end{array} \right|_{2^{n+1}+1} \qquad (12)$$

so that L_4 can be computed from L'_4 and X_h as

$$\left| \begin{array}{lr} \overline{x}_{h,0}l'_{4,(n+1)}0\overline{x}_{h,(2n-1)}\dots\overline{x}_{h,(n+2)} & + \\ x_{h,(n+1)}\dots x_{h,1} & + \\ (l'_{4,n}\vee l'_{4,(n+1)})(l'_{4,(n-1)}\vee l'_{4,(n+1)})l'_{4,(n-2)}\dots l'_{4,0} & \end{array} \right|_{2^{n+1}+1}$$

where $l_{4,i}$, $l'_{4,i}$, $x_{h,i}$ are respective bits of L_4, L'_4, X_h.

Consequently, both L'_4 and L_4 are then computed in a cascade of two 3-operand MOMAs mod $2^{n+1}+1$, each composed of a single layer of CSA followed by the 2-operand adder mod $2^{n+1}+1$. Since L'_4 does not depend on X_h, they can be computed in parallel, reducing the computations on the critical path to the 3-operand addition mod $2^{n+1}+1$.

2) Calculation of R_4: Applying the technique proposed in Section 3.2, we obtain the following expression for R_4

$$\left| \begin{array}{lr} CLN_{(n+1)}(\overline{L''_4},1)CLN_{(n+1)}(L''_4,4) & + \\ \sum_{k=1}^{\frac{n-5}{2}} CLN_{(n+1)}(L''_4,n-2k) & + \\ CLN_{(n+1)}(L''_4,n) & + \\ l_{4,(n+1)}\cdot(-2M_1^{-1})+\bar{l}_{4,(n+1)}\cdot(-M_1^{-1}) & \end{array} \right|_{2^{n+1}+1}$$

Consequently, we calculate R_4 as an $\frac{n+1}{2}$-operand addition mod $2^{n+1}+1$ of terms constructed by cyclically shifting and complementing n least significant bits of L_4 and the corrective term computed from the bit $l_{4,(n+1)}$.

3) Calculation of L_5: Similarly to the calculation of L_4, we rewrite L_5 from Equation (5) as

$$L_5 = |x_5 - X_1 - 2^{3n}R_4 + 2^nR_4|_{2^{n-1}+1}$$

Because $|2^{3n}|_{2^{n-1}+1} = -8$ and $|2^n|_{2^{n-1}+1} = -2$, therefore

$$\begin{aligned} L_5 &= |6R_4 + x_5 - X_1|_{2^{n-1}+1} \\ &= |6R_4 + \underbrace{x_5 - 2^nx_2 + X_h}_{L'_5}|_{2^{n-1}+1} \end{aligned} \qquad (13)$$

Observe now that: R_4 is an $(n+2)$-bit wide integer mod $2^{n+1}+1$; X_1 is $3n$-bit wide integer mod $2^n(2^{2n}-1)$; and x_5 is an n-bit wide operand mod $2^{n-1}+1$. A direct computation of (13) in a mod $2^{n-1}+1$ CSA tree with the necessary constant correction requires chopping these vectors into 11 operands. Again, we leave on the critical path only the computations that depend on R_4, and compute L'_5 separately in parallel, compressing it with some bit-level manipulations to

$$L'_5 = \left| \begin{array}{lr} (x_{5,(n-2)}|x_{5,(n-1)})\dots x_{5,0} + x_{2,(n-2)}\dots x_{2,0} & + \\ \overline{x}_{h,(n-3)}\dots\overline{x}_{h,(0)}\overline{x}_{2,(n-1)} & + \\ x_{h,(2n-4)}\dots x_{h,(n-2)} & + \\ x_{5,(n-1)}0\dots0\overline{x}_{h,(2n-1)}\dots\overline{x}_{h,(2n-3)} - 80 & \end{array} \right|_{2^{n-1}+1}$$

where $x_{5,i}$, $x_{2,i}$, $x_{h,i}$ are respective bits of x_5, x_2, X_h.

Once L'_5 is computed, we compute L_5 as

$$\left| \begin{array}{lr} l'_{5,(n-2)}l'_{5,(n-3)}\dots l'_{5,0} + \overline{r}_{4,(n-3)}\dots\overline{r}_{4,0}r_{4,(n-2)} & + \\ r_{4,(n-5)}\dots r_{4,1}(r_{4,0}|r_{4,(n+1)})\overline{r}_{4,(n-2)}\overline{r}_{4,(n-3)}\overline{r}_{4,(n-4)} & + \\ 0\dots0\overline{r}_{4,(n+1)}\overline{r}_{4,n}\overline{r}_{4,(n-1)}r_{4,n}r_{4,(n-1)}\overline{l'}_{5,(n-1)} & \end{array} \right|_{2^{n-1}+1}$$

where $r_{4,i}$ and $l'_{5,i}$ are the respective bits of R_4 and L'_5.

4) Calculation of R_5: By applying again the technique from Section 3.2 to each of the variants of the multiplicative inverse M_2^{-1} from Figure 1, we obtain three alternative equations to calculate R_5.

For $n = 6k+1$, $R_5 =$

$$\left| \begin{array}{lr} CLN_{(n-1)}(L''_5,n-2) + CLN_{(n-1)}(L''_5,n-3) + & \\ \sum_{i=0}^{k-2} CLN_{(n-1)}(\overline{L''_5},6i+7) & + \\ \sum_{i=0}^{k-2} CLN_{(n-1)}(L''_5,6i+4) & + \\ \overline{L''_5} + l_{5,(n+1)}| - M_2^{-1}|_{2^{n-1}+1} & \end{array} \right|_{2^{n-1}+1}$$

For $n = 6k+3$, $R_5 =$

$$\left| \begin{array}{lr} CLN_{(n-1)}(L''_5,n-3) + CLN_{(n-1)}(L''_5,n-4) + & \\ \sum_{i=0}^{k-1} CLN_{(n-1)}(L''_5,6i+2) & + \\ \sum_{i=0}^{k-2} CLN_{(n-1)}(\overline{L''_5},6i+5) & + \\ l_{5,(n+1)}| - M_2^{-1}|_{2^{n-1}+1} & \end{array} \right|_{2^{n-1}+1}$$

For $n = 6k+5$, $R_5 =$

$$\left| \begin{array}{lr} \sum_{i=0}^{k} CLN_{(n-1)}(L''_5,6i) & + \\ \sum_{i=0}^{k-1} CLN_{(n-1)}(\overline{L''_5},6i+3) + & \\ l_{5,(n+1)}| - M_2^{-1}|_{2^{n-1}+1} & \end{array} \right|_{2^{n-1}+1}$$

3.4 Final Multiplication and Addition

To compute the value of X, we rewrite (4) and (5) as

$$\begin{aligned} &x_2 + 2^nX_h + 2^n(2^{2n}-1)R_4 + 2^n(2^{2n}-1)(2^{n+1}+1)R_5 \\ =&x_2 + 2^n\underbrace{\big(\underbrace{X_h + (2^{2n}-1)R_4}_{V_1} + \underbrace{(2^{2n}-1)(2^{n+1}+1)R_5}_{V_2}\big)}_{V}. \end{aligned}$$

Again, only V_2 depends on the critical path component R_5, while the calculation of V_1 from previously computed X_h and R_4 can be done separately in parallel. Both V_1 and V_2 have to be calculated through a series of constant multiplications, additions, and subtractions. However, some bit level manipulations allow to integrate constants and extract critical and non-critical components into terms V'_1, V'_2 that conceptually correspond to terms V_1 and V_2, yet minimize the number of vectors for the final addition.

$$V = \underbrace{X_h + (2^{2n}-1)R_4}_{V'_1} + \underbrace{(2^{2n}-1)(2^{n+1}+1)R_5) - 1}_{V'_2} + 1$$

$$\begin{aligned} V'_1 =&0\dots0r_{4,(n+1)}\dots r_{4,0}x_{h,(2n-1)}\dots x_{h,0} \\ &+\underbrace{1\dots1}_{2n+1}0\underbrace{1\dots1}_{n-2}\overline{r}_{4,(n+1)}\dots\overline{r}_{4,0} + 1. \end{aligned}$$

$$\begin{aligned} V'_2 =&r_{5,(n-1)}\dots r_{5,0}0(r_{5,(n-1)}\vee r_{5,(n-2)})\dots(r_{5,(n-1)}\vee r_{5,0}) \\ &\|\overline{r}_{5,(n-2)}\dots\overline{r}_{5,0}1\overline{r}_{5,(n-1)}\dots\overline{r}_{5,0}, \end{aligned}$$

Table 1: Left: the hardware complexity estimation of the new converter. Right: the minimum delay (M. del.), area, power and power-delay-product (PDP) at the minimum delay as a function of n

Resource	[3]	This work
FA, $n = 6k - 2$	$(5n^2 + 44n - 4)/6$	
FA, $n = 6k$	$(5n^2 + 52n - 12)/6$	-
FA, $n = 6k + 2$	$(5n^2 + 48n - 8)/6$	
FA, $n = 6k + 1$		$(5n^2 + 35n - 4)/6$
FA, $n = 6k + 3$	-	$(5n^2 + 31n + 18)/6$
FA, $n = 6k + 5$		$(5n^2 + 27n + 22)/6$
HA	-	$(5n + 23)/2$
mod $2^{n-1} - 1$	2	3
mod $2^{n+1} - 1$	2	3
mod $2^{2n} - 1$	1	1
$(3n - 1)$-bit sub	1	-
$(3n + 1)$-bit sub	1	-
$4n$-bit add	1	2
Inv., n even	$6n + 1$	-
Inv., $n = 6k + 1$		$(5n^2 + 150n + 65)/12$
Inv., $n = 6k + 3$	-	$(5n^2 + 146n - 3)/12$
Inv., $n = 6k + 5$		$(5n^2 + 130n + 65)/12$
Misc.		$(n - 5)$ OR, 1 MUX

	M. del. [ns]		Area [μm^2]		Power [mW]		PDP [mW·ns]	
n	[3]	New	[3]	New	[3]	New	[3]	New
6	5.67	-	11950	-	1.88	-	10.66	-
7	-	5.83	-	14611	-	2.48	-	14.46
8	5.83	-	16114	-	3	-	17.49	-
9	-	6.28	-	19508	-	3.53	-	22.17
10	6.6	-	21550	-	4.39	-	28.97	-
11	-	6.58	-	24671	-	4.69	-	30.86
12	6.99	-	26068	-	5.98	-	41.80	-
13	-	6.82	-	30528	-	6.25	-	42.63
14	6.99	-	31657	-	6.86	-	47.95	-
15	-	6.91	-	36533	-	7.78	-	53.76
16	7.37	-	38441	-	9.12	-	67.21	-
17	-	7.37	-	43118	-	10.32	-	76.06
18	7.92	-	47588	-	12.66	-	100.27	-
19	-	7.73	-	49882	-	12.11	-	93.61
20	8.15	-	51597	-	12.97	-	105.71	-
21	-	7.72	-	57249	-	13.74	-	106.07

As a consequence, the term V_1' can be computed using a 2-operand adder with the carry-in input set to 1, whereas the term V_2' is obtained by manipulating bits of the vector R_5. The conversion result X is obtained by concatenating x_2 with the result of 2-operand addition of V_1' and V_2' in an adder with carry-in bit set to 1, which concludes the calculation. Table 1 details the complexity of the new design as compared to the design of [3].

4. EXPERIMENTAL RESULTS

Table 1 includes also the synthesis results of our design and the complementary design of [3] over the industrial 65nm low-power cell library from STMicroelectronics using Cadence RTL Compiler v8.1. We limited our comparisons to the design of [3], wherein the authors had already thoroughly compared their design to the other 5-moduli sets, and our closely matching delay, area, and power consumption trends our design fit into that comparison.

The results summarized in Table 1 also show clearly that our design techniques have alleviated the impact of the increased complexity of the new moduli set and the resulting design is competitive to the original design of [3] w.r.t. delay, area, and power, even though it has to deal with accumulation of an increased number of partial product vectors in inner product computations.

5. CONCLUSIONS

We proposed a new balanced 5-moduli RNS and its reverse converter that complements the only existing balanced 5-moduli set available for even n introduced in [3], thus bridging the 10-bit gap between consecutive dynamic ranges available for that RNS. On the example of our converter, we demonstrated three novel design techniques that allow to reduce the critical path delay of the conversion circuit, enabling the automatic synthesis tools to produce smaller and less power consuming circuits. It is important to note that many recent results on special RNSs focus on designing simple converters with little attention being paid to the complexity of the datapath's channels: the underperformance of the unbalanced channels can quickly consume any benefits not only of a simpler converter but also of RNS in general. Both the converter of [3] as well as ours are against this trend: they use well-balanced moduli sets for which not only a relatively simple reverse converter can be designed but also simple forward converters and most importantly balanced datapaths channels.

6. REFERENCES

[1] P. Ananda Mohan and A. Premkumar. 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. Circuits Syst. I*, 54(6):1245–1254, June 2007.

[2] B. Cao, C.-H. Chang, and T. Srikanthan. New efficient residue-to-binary converters for 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$. In *Proc. ISCAS*, volume 4, pages IV536–IV539, 2003.

[3] B. Cao, C.-H. Chang, and T. Srikanthan. A residue-to-binary converter for a new five-moduli set. *IEEE Trans. Circuits Syst. I*, 54(5):1041–1049, May 2007.

[4] A. Dhurkadas. Comments on a high speed realization of a residue to binary number system converter. *IEEE Trans. Circuits Syst. II*, 45(3):446–447, Mar. 1998.

[5] M. Esmaeildoust, K. Navi, and M. Taheri. High speed reverse converter for new five-moduli set $\{2^n, 2^{2n+1} - 1, 2^{n/2} - 1, 2^{n/2} + 1, 2^n + 1\}$. *IEICE Electron. Expr.*, 7(3):118–125, 2010.

[6] A. Molahosseini, C. Dadkhah, and K. Navi. A new five-moduli set for efficient hardware implementation of the reverse converter. *IEICE Electron. Expr.*, 6(14):1006–1012, 2009.

[7] A. Omondi and B. Premkumar. *Residue Number Systems: Theory and Implementation.* Imperial College Press, London, UK, 2007.

[8] S. Piestrak. Design of residue generators and multioperand modular adders using carry-save adders. *IEEE Trans. Comput.*, 43(1):68–77, Jan. 1994.

[9] S. J. Piestrak. High-speed realization of a residue to binary number system converter. *IEEE Trans. Circuits Syst. II*, 42(10):661–663, Oct. 1995.

[10] T. Shahana, R. James, B. Jose, K. Jacob, and S. Sasi. Performance analysis of FIR digital filter design: RNS versus traditional. In *Proc. Int. Symp. Commun. & Inf. Techn.*, pages 1–5, 17–19 Oct. 2007.

[11] T. Toivonen and J. Heikkila. Video filtering with Fermat number theoretic transforms using residue number system. *IEEE Trans. Circuits Syst. Video Technol.*, 16(1):92–101, Jan. 2006.