Multiplier-based Binary-to-RNS Converter Modulo \(\{2n \pm k\}\)

Pedro Miguens Matutino
ISEL/INESC-ID/IST Lisbon, Portugal
pmiguens@deetc.isel.pt
Hector Pettenghi, Ricardo Chaves and Leonel Sousa
IST/INESC-ID Lisbon, Portugal
(hector@fc.ul.pt@ips.isesc-id.pt)

Abstract—In this paper a new ROM-less structure for binary-to-RNS conversion modulo \(\{2^n \pm k\}\) is proposed. The proposed structure is based on adders and constant multipliers units without the need for ROMs. The development of this generic structure is motivated by the inefficient performance and area occupation of structures that are generated for larger Dynamic Ranges, from the existing ROM based conversion structures for modulo \(\{2^n \pm k\}\). Experimental results obtained for an ASIC technology suggest that the proposed conversion structures are on average 6 times faster and require 91% less area resources when compared with the residue converters topologies known to date.

I. INTRODUCTION

The modular characteristics of Residue Number System (RNS) offers the potential for high-speed, parallel arithmetic since that is a carry-free arithmetic system [1]. RNS is a non-weighted numbering system, which uses remainders to represent numbers. The basic arithmetic operations (add, subtract, and multiply) are easily implemented in RNS and performed over operands that are significantly shorter than the resulting RNS Dynamic Range. Typical applications for RNS are in Digital Signal Processing (DSP) for filtering, convolutions, correlations, and FFT computations [2]–[6].

The state of the art is mainly focused on RNS processors using co-prime numbers to form the three moduli set \(\{2^n - 1, 2^n, 2^n + 1\}\) [7]. DSPs based on the binary system have a Dynamic Range of \(M_{\text{bin}} = 2^n - 1\), when considering an implementation of an equivalent RNS processor, with this traditional moduli set the dimension of \(n\), has to be \(n = \lceil \log_2(2^n) \rceil = \lceil n/3 \rceil\). The Dynamic Range of this RNS processor is given by \(M_{\text{RNS}} = (2^n - 1) \cdot (2^n) \cdot (2^n + 1) = 2^{3n} - 2^n\).

Different moduli sets for the RNS processors have been proposed, in order to increase the Dynamic Range or to reduce the width of RNS channels, for example \(\{2^n \pm 1, 2^n, 2^n + 1\}\) [8], and \(\{2^n \pm 1, 2^{3n}\}\) [9], with \(m = 4n\) bit representation.

Nevertheless, the choice of moduli sets with modulo \(\{2^n \pm k\}\) channels, with unrestricted \(k\) values, are quite useful in the definition of well balanced and large Dynamic Range RNS. In addition, for a specific Dynamic Range, the choice of large RNS moduli set would result in circuits with better performance. This is due to the fact that each channel provides operands with a smaller number of bits, thus reducing the delay of the entire RNS processor.

Currently one of the most efficient binary-to-RNS converter modulo \(\{2^n \pm k\}\) is implemented using the periodic properties of powers of two of modulo \(\{2^n \pm k\}\) [10], herein denoted as \((2^n)\{2^n \pm k\}\). However, this approach provides a ROM-based topology for the converters modulo \(\{2^n \pm k\}\). In fact, all the topologies known to date will derive into unfeasible performance implementation for medium and large values of \(n\).

In this work, a novel ROM-less based generic binary-to-RNS converters for RNS with a Dynamic Range of \(m = 4n\) using \(\{2^n \pm k\}\) channels is proposed. This topology splits the \(4n\) input bits into 4 input-sets in order to multiply each input-set by its corresponding factor and reducing by using addition operations. Furthermore, the use of constant multipliers in the proposed scheme does not exhibit the exponential increment as do the structure proposed in the related state of the art. In fact, implementation results suggest, improvements of 91% in area and 6 times faster when compared with the modulo \(\{2^n \pm k\}\) RNS converters known to date.

This paper is organized as follows. Section II introduces the formulation needed to design the converters modulo \(\{2^n \pm k\}\) described in Section III. Section IV presents the experimental results of the obtained converters implemented in a 0.13\(\mu\)m Standard Cell ASIC technology from UMC. Conclusions for this work are presented in Section V.

II. FORMULATION OF RNS CONVERTERS MODULO \(\{2^n \pm k\}\)

In this section, two different ways to derive the computation of \((X)_{2^n \pm k}\) are detailed.

A. Conversion from binary-to-RNS based on periodicity

A converter modulo \(\{2^n \pm k\}\) transforms a integer \(X\) with \(m\)-bit inputs, \(\{x_{m-1}, ..., x_1, x_0\}\), into a residue word \(R\) of \(w\)-bit outputs, \(\{r_{w-1}, ..., r_1, r_0\}\), where \(w = \lceil \log_2(2^n \pm k) \rceil\), typically \(w = n\) and \(w = n + 1\) for modulo \(\{2^n - k\}\) and \(\{2^n + k\}\) respectively. The input value \(X\) is converted from \(X = \sum_{j=0}^{m-1} 2^j \cdot x_j\) into \((X)_{2^n \pm k} = \sum_{j=0}^{m-1} 2^j \cdot r_j\). In other words, it computes:

\[
(X)_{2^n \pm k} = \sum_{j=0}^{m-1} 2^j \cdot x_j / 2^{n \pm k}.
\]

(1)

However, to compute the division of \(X\) by \(2^n \pm k\), in order to find the remainder, is a hard operation. The technique proposed
in [1] give us an improved algorithm based on the expression:

\[ (X)_{2^m+k} = \sum_{j=0}^{m-1} (2^j) x_j \mod 2^m \]  

(2)

If the weight values, \((2^j)\), associated to the inputs are directly available, \((X)_{2^m+k}\) can be computed by merely adding these weight terms \((2^j)\) for which \(x_j = 1\).

The weight-selection of the inputs, \(x_j\), will be of key importance for designing the proposed converters modulo \(\{2^n \pm k\}\). From definitions based on the periodic properties of \((2^j)\) adapted from [11] can be derived two weight-selection of the inputs for \(0 \leq j \leq m-1\).

**Definition 1:** The period \(P(2^m \pm k)\) of the odd modulo \(\{2^n \pm k\}\) is the minimum distance between two distinct 1’s in the array of \(m\)-residues of \((2^j)\), i.e., \(P(2^m \pm k) = \min [j | m > j > 0, (2^j)_{2^m+k} = 1]\). \(P(2^m \pm k)\) is simply called the period of \(\{2^n \pm k\}\) [10].

**Definition 2:** The period \(HP(2^m \pm k)\) of the odd modulo \(\{2^n \pm k\}\) is the minimum distance between a pair of subsequent 1 and \(2^n \pm k - 1\) in the array of \(m\)-residues of \((2^j)\). Note that \((2^j \pm k - 1)_{2^m \pm k} = -1\) [11]. While \(P(2^m \pm k)\) exists for any \(2^m \pm k\), \(HP(2^m \pm k)\) only exists for some \(2^m \pm k\).

\(HP(2^m \pm k)\) is called a half-period because if it exists then \(P(2^m \pm k) = 2HP(2^m \pm k)\).

Let us denote the two ways of weight-selection, that can be derived from these definitions, as case 1 and 2 based on \(P(2^m \pm k)\) and \(HP(2^m \pm k)\) respectively, with \(0 \leq i \leq 2^n \pm k - 1\) for case 1, and \(-2^n \pm k + 1 \leq i \leq 2^n \pm k - 1\) for case 2. The range of the residue converter needs to be standardized in case 2, by means of an addition of a correction term \(COR\), in order to obtain residue values in the desired range \(0 \leq |i| \leq 2^n \pm k - 1\), as is directly obtained in case 1. If the inputs with associated negative weights are complemented, it is possible to chose a \(COR\) value which standardizes the range to \(0 \leq |i| \leq 2^n \pm k - 1\) [10]. Therefore, the correction value \(COR\) is chosen as the minimum value that satisfies:

\[
\begin{align*}
\left\{ \sum_{j \in (2^j)_{2^m+k} > 0} (2^j) x_j + \right. \\
\left. + \sum_{j \in (2^j)_{2^m+k} < 0} (2^j) x_j + COR \right\} & \mod 2^m = 0
\end{align*}
\]

(3)

This can be computed by setting the inputs \((x_{m-1}, ..., x_0)\) to 0 and obtaining the value of \(COR\) that provides the correct output.

**B. Conversion from binary-to-RNS based on arithmetics**

Considering a binary representation of \(X\), with \(4m\)-bit of Dynamic Range, it is necessary to compute (4) in order to obtain the residue modulo \(\{2^n - k\}\) of \(X\).

\[
\begin{align*}
(X)_{2^m+k} &= (2^{3n} X_{[n-1:3n]} + 2^{2n} X_{[n-1:2n]} + \\
&+ 2^n X_{[n-1]} + X_{[n-1:0]} \right\mod 2^m \\
&= (2^{3n} X_3 + 2^{2n} X_2 + 2^n X_1 + X_0)_{2^m+k} \\
&= (kX_3 + k^2 X_2 + kX_1 + X_0)_{2^m+k}
\end{align*}
\]

(4)

Where \(X_{[k:t]}\) represents the bits \(t\) to \(k\) of the integer \(X\).

Identically, the residue calculation modulo \(\{2^n + k\}\), is obtained by:

\[
\begin{align*}
(X)_{2^m+k} &= (2^{3n} X_{[n-1:3n]} + 2^{2n} X_{[n-1:2n]} + \\
&+ 2^n X_{[n-1]} + X_{[n-1:0]} \right\mod 2^m \\
&= (2^{3n} X_3 + 2^{2n} X_2 + 2^n X_1 + X_0)_{2^m+k} \\
&= (kX_3 + k^2 X_2 + kX_1 + X_0)_{2^m+k}
\end{align*}
\]

(5)

Taking into consideration the particular cases of modulo \(\{2^n - 1\}\) and \(\{2^n + 1\}\), conversion from binary-to-RNS can be performed by:

\[
\begin{align*}
(X)_{2^m-1} &= (N_3 + N_2 + N_1 + N_0)_{2^m-1} \\
(X)_{2^m+1} &= (-N_3 + N_2 - N_1 + N_0)_{2^m+1}
\end{align*}
\]

(6)

For these two moduli, memoryless architectures which provide fast implementations from the weighted binary-to-RNS have been described in [9] and [12].

To compute \((X)_{2^m+k}\) modular substractors are required. Nevertheless, (5) can be simplified in order to only use addition operations, as depicted in (8).

\[
\begin{align*}
(X)_{2^m+k} &= (-kX_3 + k^2 X_2 - kX_1 + X_0)_{2^m+k} \\
&= (k^3 X_3 + k^2 X_2 + kX_1 + X_0)_{2^m+k} \\
&= (k^3 X_3 + k^2 X_2 + kX_1 + X_0)_{2^m+k}
\end{align*}
\]

(8)

Since:

\[
\begin{align*}
(-X)_{2^m+k} &= (2^n - 1 - X + k + 1)_{2^m+k} \\
&= (X + k + 1)_{2^m+k}
\end{align*}
\]

(9)

were:

\[
k^3 (k + 1) + (k + 1)_{2^m+k}
\]

(10)

Considering the particular case of \((2^n+1)\) modulo equation (7) can be rewritten as:

\[
\begin{align*}
(X)_{2^m+1} &= (N_3 + N_2 + N_1 + N_0 + 4)_{2^m+1}
\end{align*}
\]

(11)

**III. RNS CONVERTER ARCHITECTURES**

In this section, the most efficient generic converters modulo \(\{2^n \pm k\}\) presented in the related state of the art and the proposed structures are described.

**A. Related work in converters modulo \(\{2^n \pm k\}\)**

Efficient parallel architectures of converters which are suited to cover the whole spectrum of \(\{2^m \pm k\}\) are presented in [10]. The first scheme is based on three levels as is shown in the Figure 1(a), which can be described by means the next four steps.

**Step 1:** The chosen weight-selection of the input bits \((x_{m-1}, ..., x_0)\) will be case 1 or case 2 depending if the modulo \(\{2^n \pm k\}\) is \(P(2^n \pm k)\) or \(HP(2^n \pm k)\), respectively.
Step 2: If \( m < 2P(2^n \pm k) \) (or \( H(2^n \pm k) + 1 \), is assumed that \( m' = m \) and go to Step 3. Otherwise, compress \( m \) input bits to \( m' \) using a CSA with End-Around Carry (EAC) of length up to \( P(2^n \pm k) \) (or \( H(2^n \pm k) + 1 \)).

Step 3: Compress \( m' \) bits into \( m'' = P(2^n \pm k) \) (or \( H(2^n \pm k) + 1 \) using a \( P(2^n \pm k)(or \( H(2^n \pm k) + 1 \)) bit CPA with EAC.

Step 4: Due to the \( \{x_{m'-1}, \ldots, x_{m}, x_{w} \} \) outputs are not in residue modulo \( 2^n \pm k \), a Final Converter (FC) ROM-based is required to compute the modulo of the resulting EAC-addition.

The FC modulo \( 2^n \pm k \) depicted in Figure 1(c), was presented in [10] with the functionality described as follows. The set of \( m'' \) inputs is partitioned into two disjoint subsets \( X_1 = \{x_{m''-1}, \ldots, x_{w}, x_{w-1} \} \) and \( X_2 = \{x_{w-2}, \ldots, x_{0} \} \), so that \( X_2 = \sum_{j=0}^{w-1} 2^j \cdot x_j \) is a residue modulo \( 2^n \pm k \) without the necessity of any more modulo computation. \( X_1 \) is converted in parallel into \( \{X_1 \}_{2^n \pm k} = \left( \sum_{j=w}^{m''-1} 2^j \cdot x_j \right)_{2^n \pm k} \) and \( \{X_1 \}_{2^n \pm k} = \left( \sum_{j=0}^{w-1} 2^j \cdot x_j \right)_{2^n \pm k} \), which is carried out by means of two \( g(m''-w+1) \times (w+1) \) ROMs. At the end, \( \{X \}_{2^n \pm k} \) is computed according to:

\[
\{X \}_{2^n \pm k} = \begin{cases} 
\{X_1 \}_{2^n \pm k} + X_2^2, & \text{if } C_{\text{out}} = 0, \\
\{X_1 \}_{2^n \pm k} - 2^n \mp k + X_2^2, & \text{if } C_{\text{out}} = 1,
\end{cases}
\]

by taking into account the least significant bits (LSBs) only.

It is important to emphasize that for large values of \( m'' \), the use of ROMs in this FC is too costly. This situation is majority common when the value of \( n \) is larger, due to in general \( m < H(2^n \pm k) \) (or \( H(2^n \pm k) + 1 \)), every coefficient \( \{2^j \}_{2^n \pm k}, 0 \leq j \leq m-1 \), represents a different residue modulo \( 2^n \pm k \) without the necessity of the \( \text{COR} \) addition. Thus, the FC-inputs for the scheme presented in Figure 1(a) is \( m'' = m \), resulting in an unpractised performance for large values of \( n \).

Therefore, for the cases where the use of large input-OMs are required, a second scheme of residue converter modulo \( 2^n \pm k \) was proposed in [10]. The scheme translates the group of bits, \( x_{w}, \) for which \( \{2^j \}_{2^n \pm k} \) are not powers of two (herein simply denoted as \( x^* \)) into binary, in order to reduce the number of input bits of the FC. The use of only one ROM to carry out the translation into binary provides a high hardware cost for large Dynamic Ranges. Thus, the computation is split into several ROMs as proposed in [10] in order to optimize the design.

Figure 1(b) shows this second topology for the particular case of \( m < H(2^n \pm k) < P(2^n \pm k) \). Essentially consist in a stage of ROMs, a stage of CSA+CPA and a stage with a FC.

The CSA+CPA stage computes the summation of the ROM-outputs and the inputs \( \{x_{m'-1}, \ldots, x_{1}, x_{0} \} \). The maximum value for the summation of the FC-inputs provides a word of \( \tau \) bits, which is smaller than the parameter \( m'' \) derived from the Figure 1(a). Therefore, the cost of the FC is reduced with this structure since \( \tau < m'' \). The last stage of the FC consist of an adder modulo \( 2^n \pm k \).

B. Proposed converter modulo \( 2^n \pm k \)

Our proposal is to implement the binary-to-RNS converters only with arithmetic units, using constant multipliers and adders units. This is a possible alternative to optimize the described ROMs implementation in order to minimize the area and performance cost of the generic converters, for larger Dynamic Ranges.

---

Fig. 1. Modulo \( \{2^n \pm k \} \) converters proposed in [10]
Two identical structures are proposed, one for modulo \( \{2^n - k\} \) and another for modulo \( \{2^n + k\} \), given that representation values on each modulo have \( n\)-bit and \( n + 1 \)-bit, respectively. In order to implement optimized structures it is considered a restriction to the \( k \) value, namely:

\[
p = \left\lfloor \log_2 k \right\rfloor \leq \frac{n}{2}.
\]

Considering (4) and the restriction of \( k \), the conversion from binary-to-RNS modulo \( \{2^n - k\} \) can be computed as:

\[
(X)_{2^n-k} = \left( k^3 X_3 + k^2 X_2 + k X_1 + X_0 \right)_{2^n-k} = \left( k^3 \right)_{2^n-k} X_3 + k^2 X_2 + k X_1 + X_0 = 2^{n-1} X_3 + 2^{n-2} X_2 + \ldots + 2^0 X.
\]

Where the constant value given by \( k^3 \) could be greater than \( 2^n - k \), since \( k < 2^5 \) and \( k^3 < 2^{2n} \), in this case the constant multiplication can be pre-computed providing an \( n \) bit array. Also, in the case of \( k^3 \) the constant could be greater than \( 2^n - k \). This value is also reduced to the modulo residue.

Considering \( m^j \) as the \( k \) constant multiplication by the addition vectors \( S^j \), of previous reduction step \( j \). In the first stage \( S^0 \) is directly \( X \) deriving:

\[
\begin{align*}
m_{3,0} & = \left( k^3 \right)_{2^n-k} X_3 \\
m_{2,0} & = k^2 X_2 \\
m_{1,0} & = k X_1 \\
m_{0,0} & = X.
\end{align*}
\]

\[
(X)_{2^n-k} = \left( m_{3,0} \right)_{2^n-k} X_3 + m_{2,0} X_2 + m_{1,0} X_1 + m_{0,0} X = S^0_{2^n-k}.
\]

The addition result \( S^1 \) obtained in (15) is not reduced modulo \( \{2^n - k\} \), thus other reduction steps are needed to calculate the final residue:

\[
(X)_{2^n-k} = \left( 2^{n-1} S^1 \right)_{2^n-k} X_3 + m_{2,0} X_2 + m_{1,0} X_1 + m_{0,0} X = 2^n S^1_{2^n-k}.
\]

From (16) it can be concluded that the converter herein proposed is implemented with three binary adders, six constant multipliers, and one modular adder. The resulting structure is depicted in Figure 2(a). This structure can be further optimized when \( n \geq 2p + 1 \), given that this optimization allows to eliminate one binary adder and one constant multiplier in the converter structure, reducing the third reduction step, as depicted in Figure 2(a) in grey color.

The conversion from binary-to-RNS modulo \( \{2^n + k\} \) has similar implementation as modulo \( \{2^n - k\} \), described in (17).
\[
\begin{align*}
\langle X \rangle_{2^n + k} &= \langle (k^3)_{2^n + k} + X_3 + k^2 X_2 + k X_1 + X_0 + c_0 \rangle_{2^n + k} \\
&= \langle m_{2n}[0] + m_{2n-1}[0] + m_{n+p-1}[0] + m_{n-1}[0] + c_0 \rangle_{2^n + k} \\
&= \langle X \rangle_{2^n + k} \\
&= \langle k^2 S_{[2n-1]} + k \rangle_{2^n + k} \\
&= \langle m_{n+1}[0] + m_{n+p-1}[0] + m_{n-1}[0] + k(k+1) \rangle_{2^n + k} \\
&= \langle S_{n+1}[0] \rangle_{2^n + k} \\
&= \langle k S_{[n+p-1]} + k \rangle_{2^n + k} \\
&= \langle m_{n+1}[0] + m_{n-1}[0] + k \rangle_{2^n + k} \\
&= \langle k^3 (k+1) + 3k(k+1) \rangle_{2^n + k}.
\end{align*}
\]

The constant compensation \(c_0\) obtained in (10) is substituted by the constant value \(c_1\). This new value has the two other compensation values needed on the reduction steps and it is given by:

\[
\begin{align*}
c_1 &= c_0 + k(k+1) + k(k+1) \\
&= \langle k^3 (k+1) + 3k(k+1) \rangle_{2^n + k}.
\end{align*}
\]

The structure of the generic converter modulo \(\{2^n + k\}\) is implemented with two binary adders, six constant multipliers and one modular adder, this converter is depicted in Figure 2(b).

IV. EXPERIMENTAL RESULTS

In order to evaluate the performance of the binary-to-RNS converters herein presented, the topologies were described in VHDL, and implemented on an ASIC technology, namely the 0.13\mu m Standard Cell Technology from UMC. The ROMs have been implemented using the Synchronous Via-1 ROM Compiler to UMC 0.13\mu m High Speed Logic Process from Faraday. The area and delay results were obtained after synthesis with Design Vision Version A-2007.12-SP5.

The first analysis was to evaluate the behaviour of converter structures, the one proposed by Piestrak [10] and the ones herein proposed, with variation of \(k\) for different values of \(n\). The experimental results presented in this paper were obtained for \(n \in\{6,30\}\) and for \(k \in\{-15,15\}\). As expected, the structure presented by Piestrak [10] has no variation for different values of \(k\) for the same \(n\). The only exception is for 5 and 4 values (out of the 65 considered values) for modulo \(\{2^n - k\}\) and \(\{2^n + k\}\) respectively. This is due to the rest of combinations \(w < H P(2^n \pm k) < P(2^n \pm k)\). The proposed structure has variations of circuit area and time delay when increasing the value \(k\) for the same \(n\), as represented.
in Figures 3(a)- 4(a) and Figures 3(b)- 4(b), respectively for area and delay. Furthermore, as expected from the theoretical analysis when comparing the area and delay values for $k = \pm 7$ and $k = \pm 9$, the $k = \pm 9$ structures have less area occupation and are faster than the first ones. This difference can be explained by the number of one bits that the value $k$ has for each modulo, for $k = \pm 7$ the constant $k$ have three one bits and for $k = \pm 9$ only two one bits. Considering this, the implementation of a constant multiplier has less requirements in terms of circuit area and has a smaller critical path.

In the second analysis the proposed structures are compared with the ones proposed by Piestrak [10]. This analysis is made using the average value of $k$ for a fixed $n$, also for a fair comparison the proposed maximum (designated as $Proposed_{max}$) and minimum ($Proposed_{min}$) $k$-values are represented. In Figure 5(a) the experimental results for circuit area are presented. From these values it can be conclude that the proposed structures allows to reduce the circuit area requirements up to 91%, when considering values of $n$ larger than 6. Furthermore, the use of constant multipliers in the proposed structure will not presents the exponential increment exhibited in the Piestrak [10], as the obtained experimental results shows in Figure 5(a). The proposed converter has a speedup up to 6 regarding the related state of art, as depicted in Figure 5(b). Furthermore, it can be observed that the Proposed $\{2^n - k\}$ converter is slower than the Proposed $\{2^n + k\}$, given that it has one more step in the reduction process as depicted in Figure 2.

Also, $AT^2$ performance metric is presented for the range of $n$ and $k$ values used in the experimental results (see Figure 5(c)). From these results it can be concluded that our proposal achieves a better performance than the ones proposed in [10].

V. CONCLUSIONS

In this paper, generic binary-to-RNS converter units for Residue Number Systems (RNS) using modulo $\{2^n - k\}$ and $\{2^n + k\}$ channels are proposed and compared with the related art. Performance results for the proposed converters for modulo $\{2^n \pm k\}$, obtained using a 0.13 $\mu$m ASIC technology, suggest that the proposed conversion structures require an average of 91% less area resources and is 6 times faster when compared with the state of art. In conclusion, this paper proposes new, generic, and efficient implementation of ROM less conversion units for the modulo $\{2^n \pm k\}$.

ACKNOWLEDGEMENT

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multiannual funding) through the PIDDAC Program funds and by the PROTEC Program funds under the research grant SFRH / PROTEC / 49763 / 2009.

REFERENCES


