Design Space Exploration based Methodology for Residue Number System Digital Filters Implementation

Cardarilli, Gian carlo; Di Nunzio, Luca; Fazzolari, Rocco; Nannarelli, Alberto; Petricca, Massimo; Re, Marco

Published in:
IEEE Transactions on Emerging Topics in Computing

Link to article, DOI:
10.1109/TETC.2020.2997067

Publication date:
2022

Document Version
Peer reviewed version

Design Space Exploration based Methodology for Residue Number System Digital Filters Implementation

Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Alberto Nannarelli[(1)], Massimo Petricca, and Marco Re

Department of Electronics, University of Rome Tor Vergata, Rome, Italy
[(1)]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark

Abstract—In the past decades, the Residue Number System (RNS) has been adopted in DSP as an alternative to the traditional two's complement number system (TCS) because of the savings in area and power dissipation.

In this work, we first perform a comprehensive Design Space Exploration (DSE) to analyze the impact of state-of-the-art design tools and libraries on the implementation of the basic operations (i.e., addition and multiplication) used in DSP. From this DSE, we extract the characteristics of the stand-alone RNS and TCS operators in the different design corners, independently of the specific context of the application.

Then, we propose a design methodology, based on the DSE, to fully automate the design of digital filters, hiding the detail of the RNS to the designer, and providing optimal power efficient implementations. Our methodology can enable the efficiency in computation (speed and power) in DSP and in emerging applications, such as Machine Learning and Internet-of-Things.

Index Terms—Residue Number System (RNS), design space exploration, trade-offs analysis.

1 Introduction

The Residue Number System (RNS) has been extensively used over the years in digital systems as an alternative to binary representation [1], [2].

The RNS allows the parallelization of operations such as addition and multiplication, in independent datapaths with reduced bit-width. For operations on integers of large dynamic range (bit-width), the advantage is that the RNS parallelization shortens the carry chains reducing the delay of the carry propagation in the RNS channels resulting in faster execution of the operations [1].

The main drawback is that conversions are necessary to interface a RNS datapath to the conventional systems based on binary representation.

Among several application areas, the RNS is mainly used in Digital Signal Processing (DSP). In the last decades, digital filters were implemented in RNS to increase the speed and to reduce the power dissipation, e.g., [3], [4], [5], [6], [7], [8].

Another interesting field of application is the RNS implementation of fault-tolerant DSP architectures [9].

More recently, RNS has been used to optimize the hardware implementation of neural networks to lower the energy in multiply and accumulate operations in the inference computation, [10], [11], [12], [13], [14].

These research works indicate RNS as a good candidate for the implementation of energy efficient processors to be integrated in embedded systems and Internet-of-Things devices.

In this work, we analyze the impact of today's integrated circuit technology and state-of-the-art CAD tools in the design of RNS digital filters by comparing the RNS implementation to the implementation in the conventional two's complement system (TCS).

Our main objective is to show that by using synthesis and hardware generators for RNS processors, we can design energy efficient DSPs (e.g., FIR filters) and hide the detail (conversion algorithms, choice of moduli, etc.) to the designers.

The design choices for the RNS processor are based on a Design Space Exploration (DSE) to investigate the performance of the TCS and RNS arithmetic operators used in DSP (i.e., addition, multiplication, and fused multiplication-addition). We characterize each operator in terms of delay, area and power dissipation for a set of different dynamic ranges typical of DSP applications.

By the DSE, we identify design corners, for each operator, where the RNS is advantageous over the TCS. Then, we present a design methodology based on the DSE and on RNS synthesis tools to automatically generate RNS filters targeting given design constraints to validate the results of the DSE.

The main contributions of this work are:

- A comprehensive design space exploration of basic arithmetic operators where several metrics are compared between TCS and RNS implementations. These include: switching capacitance, impact of glitches and drive strength on power dissipation.

- A characterization of the switching activity in TCS and RNS units under several classes of input patterns.
- A design methodology to automatically design RNS filters to optimize power efficiency.

The paper is organized as follows. In Section 2, we present the background on RNS arithmetic and filters. In Section 3, we explain how we carried out the DSE of the arithmetic operations and we show its results. In Section 4, we highlight the main findings of the DSE. In Section 5, we illustrate how the proposed design methodology is applied to some examples to obtain power efficient digital filters. In Section 6, we report the results of related work supporting the findings of the DSE. In Section 7, we wrap-up the main findings of the work. In the Supplemental Material, we provide an appendix reporting the experimental data of the DSE.

2 Residue Number System

A Residue Number System (RNS) is defined by a set of P coprime integers $\{m_1, m_2, \ldots, m_P\}$ which identify the RNS base. Its dynamic range is given by the product

$$M = m_1 \cdot m_2 \cdot \ldots \cdot m_P \tag{1}$$

Any integer $X \in \{0, 1, 2, \ldots M-1\}$ has a unique RNS representation given by

$$X \overset{RNS}{\rightarrow} (\ \langle X \rangle_{m_1}, \langle X \rangle_{m_2}, \ldots, \langle X \rangle_{m_P}\ ) \tag{2}$$

where $\langle X \rangle_{m_i}$, or X_{m_i}, denotes the operation $X \ mod \ m_i$.

Operations, such as addition and multiplication, are computed independently (in parallel) in each modulus m_i path

$$Z = X \ op \ Y \quad \overset{RNS}{\rightarrow} \quad \begin{cases} Z_{m_1} = \langle X_{m_1} \ op \ Y_{m_1} \rangle_{m_1} \\ Z_{m_2} = \langle X_{m_2} \ op \ Y_{m_2} \rangle_{m_2} \\ \ldots \quad \ldots \quad \ldots \\ Z_{m_P} = \langle X_{m_P} \ op \ Y_{m_P} \rangle_{m_P} \end{cases} \tag{3}$$

As a consequence, operations on large wordlengths can be split into several modular operations executed in parallel and characterized by a reduced wordlength [1].

The main disadvantage of the RNS approach, when used in a TCS environment, is the need for input and output converters. The input conversion can be implemented by using a look-up table approach when the dynamic range of the moduli is small [1], or by computing the residues by extracting the modulus [15]. The conversion from the RNS representation to TCS is normally done by the Chinese Remainder Theorem (CRT) [1]

$$Z = \left\langle \sum_{i=1}^{P} \overline{m_i} \cdot \langle \overline{m_i}^{\,-1} \rangle_{m_i} \cdot Z_i \right\rangle_M \tag{4}$$

with $\overline{m_i} = \frac{M}{m_i}$ and $\overline{m_i}^{\,-1}$ obtained by $\langle \overline{m_i} \cdot \overline{m_i}^{\,-1} \rangle_{m_i} = 1$.

For example, suppose we have the RNS base $\{\ 5,\ 7,\ 8\ \}$, with dynamic range $M = 5 \cdot 7 \cdot 8 = 280$. The addition $13 + 150 = 163$ is performed in RNS as follows:

1) conversion to RNS

$$13 \overset{RNS}{\rightarrow} \begin{cases} \langle 13 \rangle_5 = 3 \\ \langle 13 \rangle_7 = 6 \\ \langle 13 \rangle_8 = 5 \end{cases} \qquad 150 \overset{RNS}{\rightarrow} \begin{cases} \langle 150 \rangle_5 = 0 \\ \langle 150 \rangle_7 = 3 \\ \langle 150 \rangle_8 = 6 \end{cases}$$

2) modular addition in parallel RNS channels

$$\begin{gathered} \langle 3 + 0 \rangle_5 = 3 \in [0, 4] \\ \langle 6 + 3 \rangle_7 = 2 \in [0, 6] \\ \langle 5 + 6 \rangle_8 = 3 \in [0, 7] \end{gathered}$$

3) conversion to TCS by (4)

$$\left.\begin{matrix} 3 \\ 2 \\ 3 \end{matrix}\right\} \overset{CRT}{\rightarrow} \langle\ 56 \cdot 1 \cdot 3 + 40 \cdot 3 \cdot 2 + 35 \cdot 3 \cdot 3\ \rangle_{280} = 163$$

with

$$\begin{gathered} \overline{m_5} = 56, \ \ \overline{m_5}^{-1} = 1 \\ \overline{m_7} = 40, \ \ \overline{m_7}^{-1} = 3 \\ \overline{m_8} = 35, \ \ \overline{m_8}^{-1} = 3 \end{gathered}$$

Furthermore, the increased complexity is also justified by the greater robustness to the errors in RNS, where a fault in a modular processor does not preclude the correct operation of the filter, but only the reduction of its dynamic range. The use of fault-tolerant by construction arithmetic circuits is very important with the shrinking of microelectronic technologies because of the increased occurence of radiation induced, faults, even at sea level [9].

2.1 Digital Filters in RNS

A Finite Impulse Response (FIR) filter is described by

$$y(n) = \sum_{k=0}^{N-1} a(k) \cdot x(n-k) \tag{5}$$

where N is the order of the filter, $x(n)$ and $y(n)$ are respectively the input and the output sequences, and $a(k)$ are the filter coefficients (filter mask).

FIR filters are largely used for the linear phase characteristic. However, for high selectivity in FIR filters , i.e., a narrow transition band, a high order N is necessary. Moreover, if the dynamic range is large and the data rate is high, a fully parallel FIR implementation is needed. An efficient parallel implementation of (5) when N is high and many bits b are required for coefficients/samples, is challenging because of the large number of bxb multipliers. In these cases, a RNS implementation of FIR filters is very interesting mainly for the reduction in complexity of the multiplication due to a reduced dynamic range of the residues, and the possibility to use the isomorphism method that transforms the multiplication in a modular sum (see Section 3.5.2).
By (3), the filter (5) is specified in RNS as

$$y(n) = \sum_{k=0}^{N-1} a(k) \cdot x(n-k) \overset{RNS}{\rightarrow}$$

$$\begin{cases} y_{m_1}(n) = \left\langle \sum_{k=0}^{N-1} \langle a_{m_1}(k) \cdot x_{m_1}(n-k) \rangle_{m_1} \right\rangle_{m_1} \\ y_{m_2}(n) = \left\langle \sum_{k=0}^{N-1} \langle a_{m_2}(k) \cdot x_{m_2}(n-k) \rangle_{m_2} \right\rangle_{m_2} \\ \qquad \ldots \\ y_{m_P}(n) = \left\langle \sum_{k=0}^{N-1} \langle a_{m_P}(k) \cdot x_{m_P}(n-k) \rangle_{m_P} \right\rangle_{m_P} \end{cases} \tag{6}$$

and its hardware architecture is shown in Fig. 1.

For illustrative purposes, we refer to real coefficients FIR filters in the transposed form [16].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

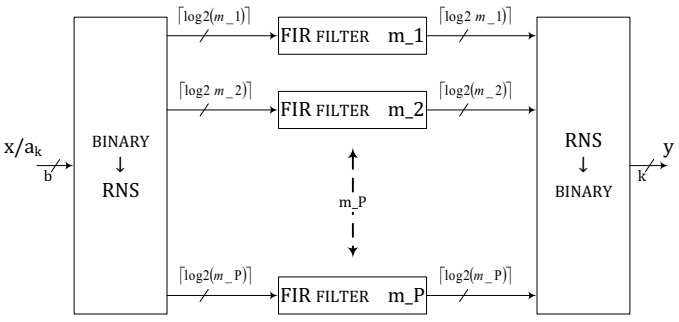

Fig. 1: RNS implementation of a FIR filter.

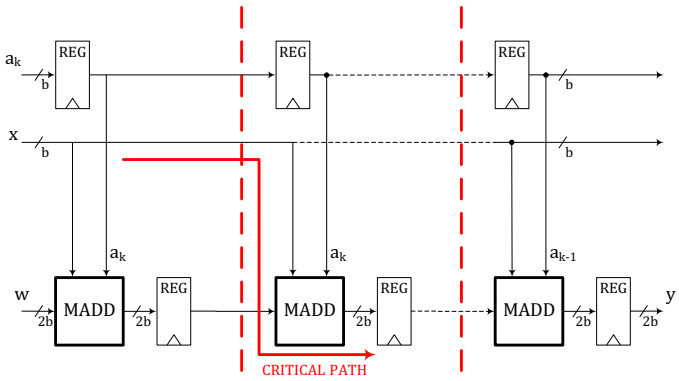

Fig. 2: TCS implementation of the transposed form real coefficients FIR filter (section).

The real coefficients FIR filter in two's complement is shown in Fig. 2 for a section of the filter. The filter is composed of a regular repetition of blocks (multiply-add and register in the figure), a so called *filter's tap*.

In RNS, the structure of the modular filter in Fig. 1 is obtained from Fig. 2 by using modular multiply-add units for each RNS channel.

The main advantage of the RNS decomposition of the FIR filter in Fig. 1 is that the modular operations in the RNS channels are done on a reduced dynamic range with respect to the dynamic range of the TCS. Consequently, the critical path is reduced and the slack with respect to the TCS implementation can be used to optimize the energy consumption.

As previously mentioned, one of the disadvantages of the RNS approach is the need for input and output converters which increase the area and the power consumption of the RNS processor. However, the power consumption in the converters does not offset the RNS benefits when a high number of operations per sample is required (e.g., the order of the filter N is high).

2.2 Choice of the RNS Base

The choice of the moduli set, or RNS base, is critical to obtain an efficient RNS implementation. From (1) we can roughly assume that the average number of bits (bit-width) of each modulus composing the RNS base is

$$d_{ave} = \frac{\sum_{i=1}^{P} \lceil \log_2 m_i \rceil}{P} \tag{7}$$

To cover a given dynamic range D (such as $D \leq M$) we can select the RNS base by using two different criteria:

1) RNS base composed of a small number P of moduli. In this case, by (7) the dynamic range of each modulus in the base is medium/large.
2) RNS base composed of a relative large number P of moduli. In this case, the dynamic range of each modulus in the base is medium/small.

By applying criterion 1) for the implementation of a typical high-performance digital filter (20–40 bits of output dynamic range), the bit-width of each base is rather large. For example, to implement a 24-bit dynamic range by using a $P = 3$ RNS base (e.g., $\{2^k - 1, 2^k, 2^k + 1\}$), we need at least $k = 9$ resulting in $d_{ave} \simeq 9$.

In contrast, by using criterion 2) we can cover the 24-bit dynamic range by $P = 6$ and the base $\{7, 11, 13, 17, 31, 32\}$ for a resulting $d_{ave} = 4.3$.

The main advantage in using several prime moduli of small dynamic range is that we can implement modular multiplication by table look-up and addition (no actual multiplication) as explained in Section 3.5.2.

Moreover, by criterion 2), we have more freedom in selecting the co-prime moduli of the RNS base at expenses of more complicated input and output converters. However, converters constitute an overhead independently of the chosen RNS base, and, generally, the complexity of the converters does not have a significant impact on applications in which the number of RNS operations (additions and multiplications) per sample is large.

In other words, if we obtain a significant gain in area, or power, in the RNS computation the overhead introduced by different conversion schemes becomes marginal.

Another criticism to the use of a RNS base composed of a large number of moduli is that, due to the difference in the moduli's bit-width, the delay in the different paths is unbalanced. However, as modern synthesis tools optimize slack for low power (e.g., selection of high-threshold voltage gates in non-critical paths), unbalanced paths might result at advantage for reducing the power dissipation. Moreover, voltage scaling techniques can be applied to increase the delays in the faster moduli (delay equalization to the slower modulus) saving additional power [5].

2.3 Coding Overhead

In [17], we introduced the *"coding overhead"* (OH) defined as the amount of extra bits required in the RNS representation of an integer compared to its TCS representation.

If the dynamic range of the TCS is $D = 2^d$, for the RNS representation the base must be chosen such that

$$\prod_{i=1}^{P} m_i = M \geq D = 2^d \tag{8}$$

The total number of bits necessary for the RNS base is

$$b = \sum_{i=1}^{P} \lceil \log_2 m_i \rceil \tag{9}$$

The coding overhead is defined as

$$OH = b - d \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

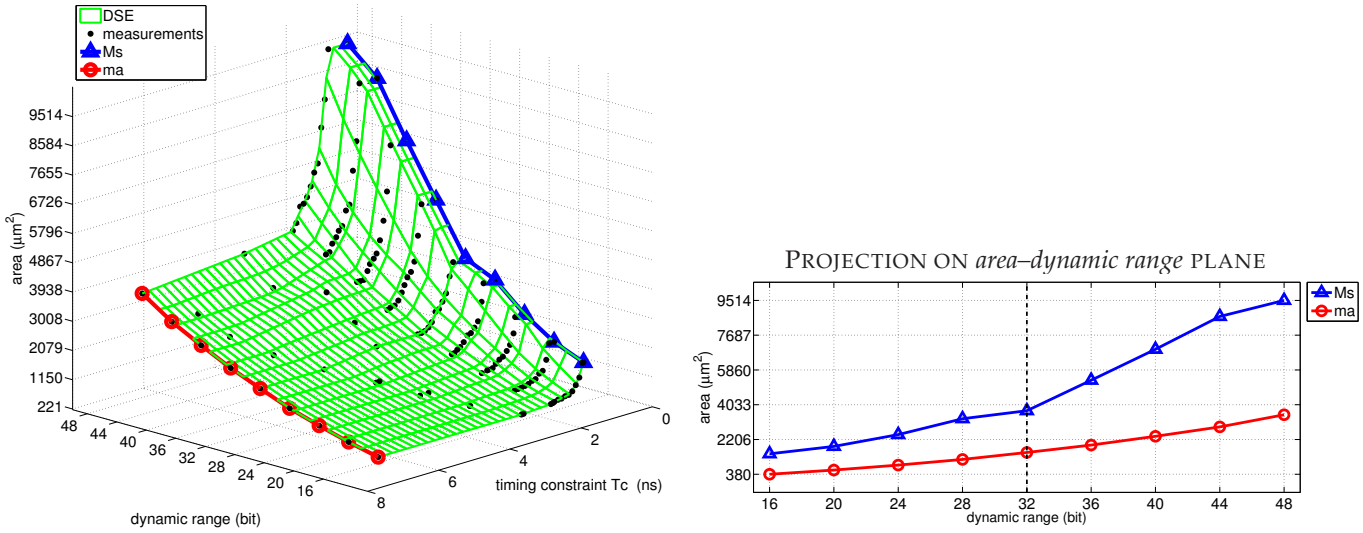

Fig. 3: TCS multiplier plots. 3D Area (left). Projection on the 2D *area–dynamic range* plane (right).

A large OH has a significant impact on registers and multiplexers, i.e., the parts of the datapath to store and move the data around. For example, in case the operands of a $n \times m$-bit multiplication (*multiplicand* and *multiplier*) need to be stored, in a TCS multiply unit the *multiplicand* storage requires n bit, and the *multiplier* m bits. In RNS, both *multiplicand* and *multiplier* must be expressed in the whole RNS base requiring $2 \times (n + m + OH)$ bits of storage[1].

3 Design Space Exploration

In this section, we first explain the DSE experimental set-up, illustrate some of the metrics used and describe some specific choices we made. Then, in the second part, we present the results of the actual DSE on the three operations: addition, multiplication, and fused multiplication and addition.

3.1 DSE Framework

The DSE is based on a synthesis design flow (Synopsys) targeting the low-power STM 45 nm library of standard cells.

The functional units implement in both TCS and RNS the following operations: addition (ADD), multiplication (MULT), and fused (merged) multiplication-addition (MADD).

The characterization is done on a set of dynamic ranges which are typical of digital filters: from 16 to 48 bits, at steps of 4 bits.

In the DSE, we consider two design corners:

- ***Ms*** the maximum speed corner, obtained by synthesizing the functional unit to achieve the maximum speed.
- ***ma*** the minimum area corner, obtained by synthesizing the unit to achieve the minimum area.

1. For example, for a 10×8 TCS multiplier, 18 flip-flops are necessary to store the two operands. In RNS, for the same multiplier even assuming $OH = 0$, a total $2 \times (18 + 0) = 36$ flip-flops are required.

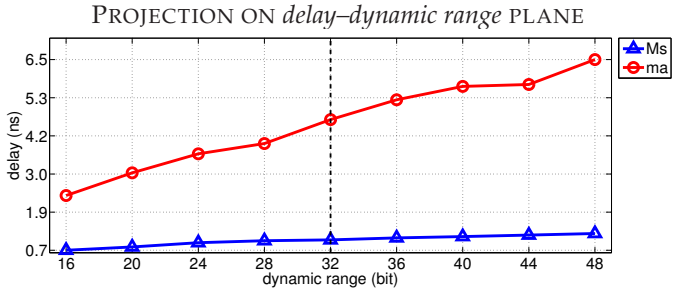

Fig. 4: Projection on the *delay–dynamic range* plane for the TCS multiplier.

For each operation and dynamic range, we perform a delay, area and power dissipation characterization for twelve different synthesis timing constraints T_C, ranging from the *ma* to the *Ms* corners.

The two corners for the area of the TCS multiplier are illustrated in Fig. 3. The 3D plot (area–dynamic range–timing constraint) is projected on the *area–dynamic range* plane in Fig. 3 (right). A similar 3D plot is obtained for delay–dynamic range–timing constraint. Its projection on the *delay–dynamic range* plane is shown in Fig. 4.

As an example, we show the case for a TCS multiplier. At 32-bit dynamic range, dashed vertical lines in Fig. 3 and Fig. 4, the *"area"* plot of Fig. 3 (right) points to an area of about $1,500\ \mu m^2$ at *ma* and about $3,500\ \mu m^2$ at *Ms*. The *"delay"* plot of Fig. 4, points to a delay of about $4.6\ ns$ at *ma* and about $1.0\ ns$ at *Ms*. Therefore, a 32-bit TCS multiplier can be synthesized to a target delay in the range $[1.0, 4.6]\ ns$ adapting its delay and area to the specified timing constraint T_C.

The power estimation is based on simulations on the synthesized netlists of the units and includes an estimate of the interconnect capacitance, based on the Synopsys' built-in models[2].

2. Synopsys Design Compiler (DC) interconnects' delays are estimated proportional to the circuit area when the layout is not available.

To determine the switching activity, we opted for a Monte-Carlo approach [18]. The random vectors are generated with a uniform distribution on the whole dynamic range of the operands.

The power dissipation is estimated at a data rate of 100 MHz for all units, since the corresponding delay (10 ns) is sufficient for signals to propagate input to output in all synthesized units.
In the DSE, the units are never pipelined.

3.2 Characterization Metrics

In the DSE, besides the characterization of the traditional metrics (delay, area and power dissipation), we extract info on the switching activity and capacitance of the circuits' internal nodes, and info on the percentage of transitions due to glitches.

3.2.1 Switching Capacitance

We analyze the combined effect of the switching activity and the capacitance in the internal nodes of the circuits under characterization.

The dynamic power dissipation of a circuit composed of N cells (gates) can be modeled by

$$P_{TOT}^{dyn} = \sum_{i=1}^{N} \left(\frac{1}{2} V_{DD}^2 C_{Li} + {E_i}^{int} \right) a_i f_C \qquad [W] \quad (11)$$

where V_{DD} is the power supply voltage, C_{Li} is the capacitative load connected at the output of the i-th gate, ${E_i}^{int}$ is the internal energy (sum of short-circuit currents and switching of internal nodes), and a_i is the switching activity of the output node of the gate expressed as percentage of the circuit toggling (clock) frequency f_C ($a_C = 1.0$). At a given supply voltage V_{DD}, the cell's power dissipation is largely due to the product $C_{Li} \cdot a_i$, called *"switching capacitance"*.

We investigate how the correlation between the two factors impacts the overall power dissipation in TCS and RNS operators.

3.2.2 Cells' Drive Strength

Standard cells libraries provide several drive strengths (current sourcing) for the same logic function. These cells are normally labeled "1X", "2X", etc., where a cell labeled 2X sources double the current sourced by a cell 1X.

Since cells with higher drive strengths consume more power, modern logic synthesizers adapt the drive strength to the load and the speed of the given portion of the circuit.

Consequently, the mix of cells' drive strength mostly depends on how well a given operator architecture can match the specific timing constraint. Since TCS and RNS architectures are quite different, especially for multiplication, the high drive cell-mix differences (and the power dissipation) are quite significant in some cases.

3.2.3 Glitches

Spurious transitions, or glitches, are mostly due to delay mismatches at gates' inputs and occur less frequently in shallower paths. The power due to the glitches, can account for a large percentage of the total [19], [20].

The glitch count is performed by computing the difference in the number of transitions, in each node, between a full-timing (with parasitics delays) and a zero-delay simulation.

3.3 Choice of RNS base for DSE

To cover a given dynamic range in RNS, we can choose several sets of co-prime moduli to form the RNS base (Section 2). The moduli considered for the RNS base are all prime numbers from 3 to 71 and at most one power-of-two (2^k) from 8 to 256.

Once we have the data from the characterization of the different moduli, we choose the set of moduli that minimizes a specific cost function. The costs are determined by a stand-alone characterization of the modular operations for each modulus. This characterization is carried out for different timing constraints from the minimum value (corresponding to the maximum speed implementation) to a very relaxed T_C value (corresponding to the minimum area implementation).

We consider all the possible combinations of co-prime moduli which cover the range D, plus other combinations with small overhead on D (e.g., d+2 bits):

$$2^d = D \leq \prod^{P} m_i = M \leq 2^{d+2} \quad (12)$$

Then, we choose the set (RNS base) which minimizes the cost function (e.g. area or power dissipation of the RNS operator) at the given timing constraint T_C.

If the delay at maximum speed for a given modulus m_i is larger than T_C, the modulus is discarded, otherwise m_i, and its corresponding cost, is added to the base. When all combinations of moduli covering the specific dynamic range have been evaluated, we select the RNS base with the lowest cost.

For example, for RNS addition and $D = 2^{16}$ a sub-set of the bases satisfying (12) at $T_C = 0.3\ ns$ is:

$$\begin{array}{lllll} \ldots & & & & \\ \text{base } i: & \{31, 47, 64\} & \longrightarrow & \text{cost} = 206 & \\ \text{base } j: & \{3, 7, 31, 128\} & \longrightarrow & \text{cost} = 148 & \leftarrow \text{ min.} \\ \text{base } k: & \{3, 5, 7, 31, 32\} & \longrightarrow & \text{cost} = 160 & \\ \ldots & & & & \end{array}$$

We choose base j which minimizes the cost function for the specific timing constraint.

3.4 DSE Addition

3.4.1 TCS Adder

The TCS adder architecture is selected by the synthesizer depending on the timing constraint[3]. For relaxed values of the time constraint the synthesizer implements a structure similar to a carry-ripple adder (CRA), where the propagation delay increases linearly with the adder bit-width. For tighter values of the time constraint the synthesizer implements a prefix-adder scheme [21], where delay increases as the logarithm of the adder bit-width.

3. Synopsys' DWare provides several adder architectures automatically selected to meet the design constraints.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

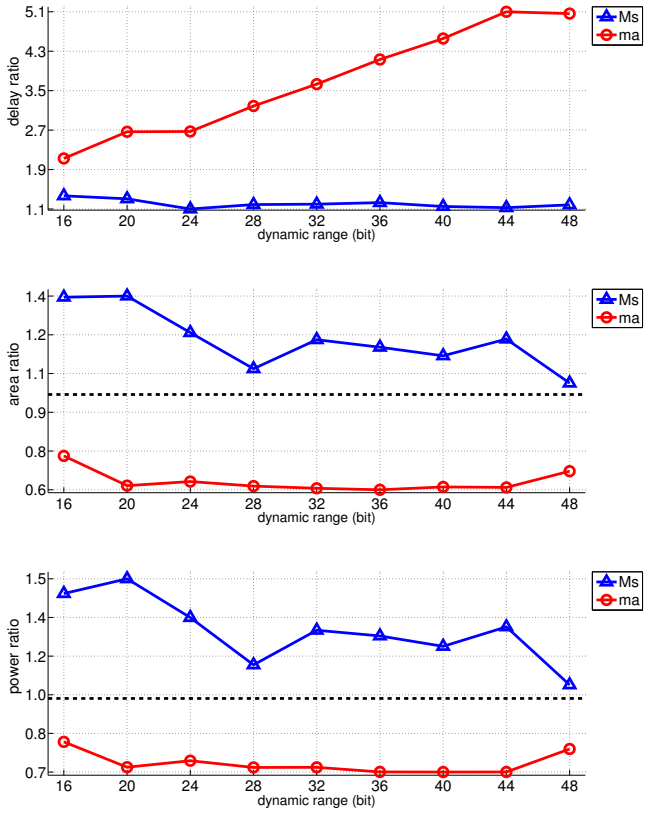

Fig. 5: DSE addition: ratios TCS/RNS in *Ms* and *ma* corners.

3.4.2 RNS Adder

The easiest way to implement two-operand modular addition $\langle a+b\rangle_m$, for any modulus m, is to perform two additions modulus 2^k (with $m < 2^k$)

$$(a+b) \quad \text{and} \quad (a+b-m) \tag{13}$$

If the result of $a+b$ exceeds the modulus, by subtracting m we obtain the correct result.

This simple algorithm can be implemented by two cascaded adders as in [22]. To reduce the latency, the two additions of (13) are executed in parallel (e.g., [23]).

From the HDL description of the modular algorithm the synthesizer selects the implementation depending on the design corner. Synthesizer's optimization criteria (timing constraint driven) apply to modular adders as well.

3.4.3 Characterization Results

Fig. 5 shows the trends of the ratio for delay, area, and power of the TCS and the RNS adder for the different dynamic ranges. The actual values for delay, area and power dissipation are reported in the Supplemental Material. Ratios are computed by placing the TCS value at the numerator. Therefore, for ratios greater than 1.0 the RNS is advantageous.

The RNS adder is always faster than the TCS adder. In the maximum speed (*Ms*) corner the speed-up (ratio) is marginal: ranging from 1.1 to 1.4. However, in the minimum area (*ma*) corner, the RNS adder is much faster (speed-up 2.1–5.1) because, in this corner, the adders are synthesized as CRA and in RNS modular paths the carry-chain is much shorter.

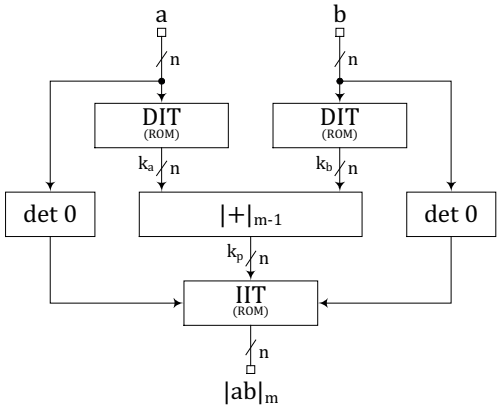

Fig. 6: Structure of basic isomorphic multiplier.

As for the area and average power dissipation, Fig. 5 (bottom plots) shows that in the *ma* corner, the addition in RNS is never advantageous as the ratios are less than 1 (marked with a dashed horizontal line) for all dynamic ranges. This result can be easily explained by considering that modular adders, require n+OH full-adders (FAs), needed for the addition, plus extra gates for the modular correction, while a minimum area n-bit TCS carry-ripple adder requires n FAs only.

For the maximum speed design corner, area and power dissipation are lower in RNS, but the improvements never exceed 40%.

In summary, in the *Ms* corner the RNS choice is always advantageous, while in the *ma* corner, the RNS is justified only if the delay is critical: i.e., the TCS adder (slower than RNS in the *ma* corner) does not meet the required timing constraint.

3.5 DSE Multiplication

3.5.1 TCS Multiplication

The TCS multiplication is implemented by a radix-4 parallel multiplier, which gives the best delay–power dissipation trade-offs [24].

3.5.2 RNS Multiplication

By using moduli which are prime numbers, we can transform a modular multiplication into a modular addition of the isomorphic indices [1].

The *isomorphism* method is similar to the method of multiplication by logarithms: the two operands are transformed into their indices of the isomorphism (logarithms) by Direct Isomorphism Transforms (DITs), the indices (logarithms) are added, and their sum is transformed back (anti-logarithm) by the Inverse Isomorphism Transform (IIT). The architecture of the isomorphic multiplier is shown in Fig. 6. The scheme is completed by a few gates to detect when one of the two operands is zero (no corresponding index in the isomorphism). On zero detected, the product is set to zero.

If the modulus' bit-width is small (less than 8 bits), the direct and inverse isomorphic transformations can be implemented by look-up tables of reasonable size which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

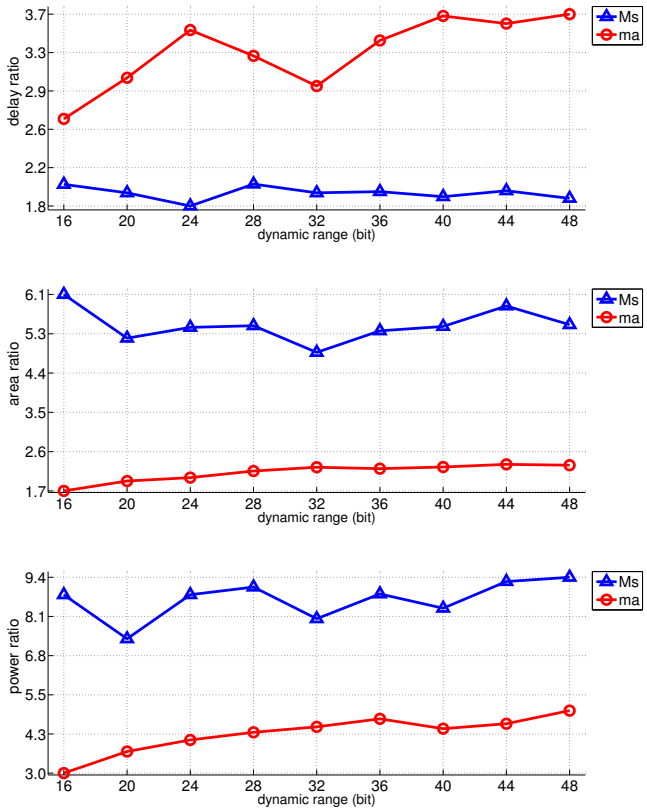

Fig. 7: DSE multiplication: ratios TCS/RNS in *Ms* and *ma* corners.

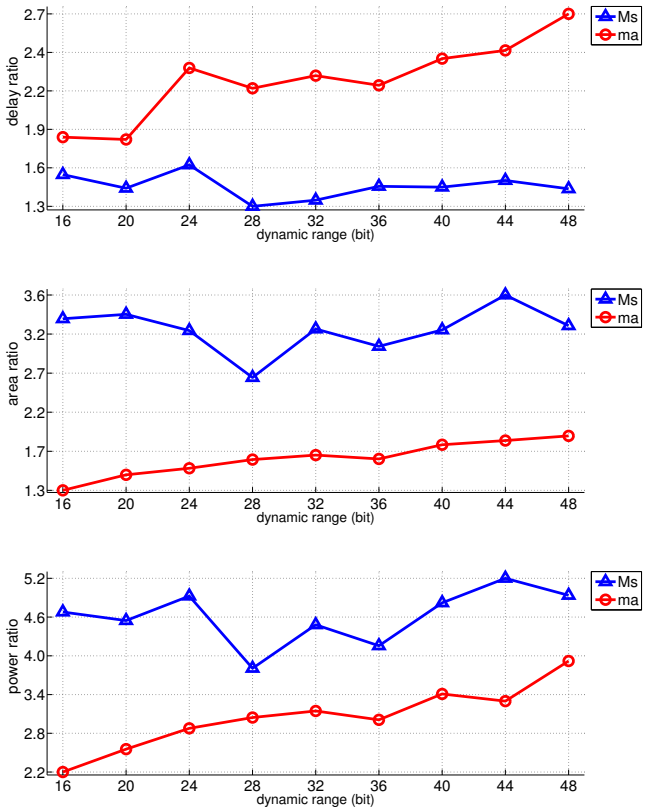

Fig. 8: DSE MADD (Multiply-Add: ratios TCS/RNS in *Ms* and *ma* corners.

are then synthesized into multi-level combinational logic in standard cells.

For the RNS isomorphic multiplication, we considered the two alternatives of [25] and [26]. Detail on the two alternatives is given in [27].

3.5.3 Characterization Results

Also for multiplications, we show in Fig. 7 the trends of the ratio TCS/RNS for delay, area and power dissipation for the different dynamic ranges. The actual values are reported in the Supplemental Material.

Differently from the addition case, RNS multipliers are always advantageous in both corners (*ma* and *Ms*).

TCS multipliers, especially for large dynamic ranges, require adder trees which are large (area) and consume a significant amount of power. In contrast, in RNS, by taking advantage of the isomorphism, the modular multipliers are changed in adders which are much smaller than binary multipliers.

By comparing the ratios of area and power in Fig. 7 (bottom plots), it is clear that the lower power dissipation of the RNS is not only due to the reduced area, but also to the reduced switching activity, as shown in detail in Section 4.

3.6 DSE Multiplication-Addition

The multiplication-addition (MADD in the following), $Z = X \cdot Y + W$, is a very frequent operation in most DSP algorithms.

3.6.1 TCS MADD

In TCS, the MADD is efficiently implemented by adding W (properly aligned) in the partial products reduction tree of the multiplier. Therefore, the performance of the TCS MADD is very similar to that of the TCS multiplier.

3.6.2 RNS MADD

On the contrary, the modular MADD architecture is very different from a modular multiplier implemented by isomorphism, because the indices calculus domain cannot be merged with the residue domain[4]. Consequently, the RNS MADD is implemented by cascading an isomorphic multiplier and a modular adder, for each modulus of the RNS base. Therefore, in this case, the performance is roughly the sum of those of modular multiplication and modular addition.

3.6.3 Characterization Results

The plots of ratios for the MADD operator DSE are shown in Fig. 8, while the DSE actual values are reported in the Supplemental Material.

In both the maximum speed (*Ms*) and the minimum area (*ma*) design corners, the RNS MADD is always faster, smaller and more power efficient than the TCS MADD, but the advantages are reduced compared to the implementation of multiplication only.

4. An inverse isomorphism transformation after multiplication is required.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

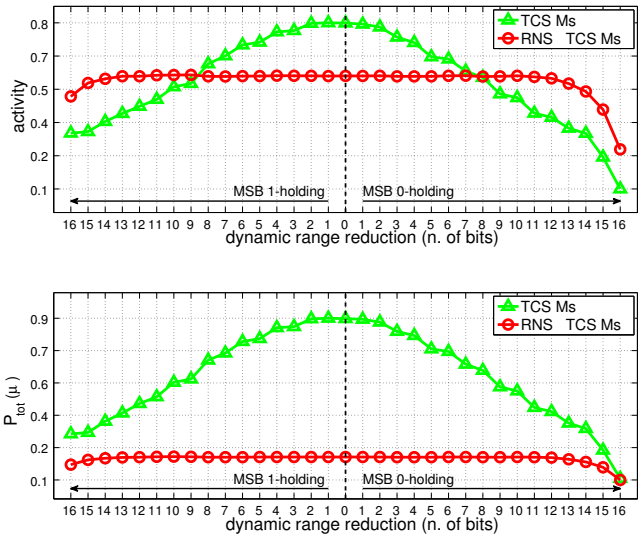

Fig. 9: Switching activity (top) and total power consumption (bottom) in 32-bit MADD when the dynamic range of test-vectors is progressively reduced.

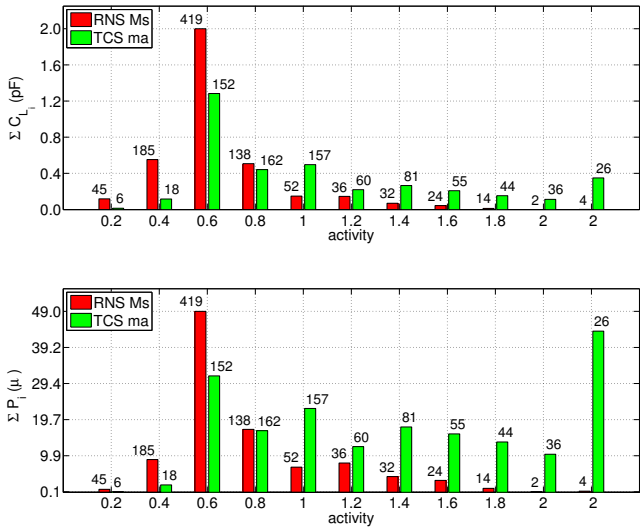

Fig. 10: Switching capacitance (top) and dynamic power dissipation (bottom) for 32-bit MADD.

Summarizing the results, the RNS MADD implementation is 30–60% faster than the corresponding TCS implementation in the *Ms* corner and about two times faster in the *ma* corner. As for the power dissipation, the RNS MADDs consume roughly one quarter of the power in the *Ms* corner and about one third of the power in the *ma* corner.

3.6.4 Switching Activity when Dynamic Range is Reduced

The characterization of the MADD is carried out by applying random test-vectors uniformly distributed in the whole dynamic range of the input signals.

However, for frequency filters, some coefficients have a very reduced dynamic range with respect to the full dynamic range. This is not the case for other types of filters, such as correlation filters.

In linear-phase frequency filters with N coefficients, the coefficients closer to the mid-point $N/2$ have a full dynamic range, while the ones closer to the end-points (i.e., 0 and $N-1$), also called *filter's tails*, have a reduced dynamic range. When we use a multiply-add unit to implement FIR filters, one of the inputs (we assume the *multiplier* – opposed to *multiplicand*) is connected to one of the coefficients specifying the filter mask (see later in Section 5). To simulate this case, we generate random test-vectors at a progressively reduced dynamic range in the input Y of the multiply-add unit ($Z = X \cdot Y + W$). For the other two inputs, X and W, we apply random test-vectors at full dynamic range.

The results of the experiment for a 32-bit MADD are shown in Fig. 9.

In Fig. 9, we indicate on the x-axis how many bits (starting from the Most-Significant Bit, MSB) are held constant (to 0 or 1). By holding the MSBs to the sign bit (MSB in TCS) we reduce the dynamic range of the specific test-vector. In Fig. 9, the point labeled '0' (center of the plot) indicates that 0 bits are held constant and that the corresponding MADD works at full dynamic range. Moving from the center of the plot toward the right, we hold an increasing number of MSBs to 0 by reducing the dynamic range for positive coefficient values. Similarly, for negative values, the range is reduced by holding the MSBs to 1 (from the center toward the left of the plot).

In Fig. 9, both the switching activity (top) and the power consumption (bottom) plots show that the TCS MADD is sensitive to the variations in the dynamic range of one of the operands (bell shaped trend), while the RNS MADD is almost insensitive to those variations.

In summary, although the data of Fig. 8 points to a better power efficiency of the RNS over the TCS MADD, the benefits are reduced in actual frequency filters since in the filter's tails the TCS MADDs have a reduced power dissipation due to the smallest dynamic range of their operands.

4 Main Findings of DSE

In the following, we list the main characteristics emerged from the design space exploration of the TCS and RNS operators.

These aspects, somewhat hidden in the power dissipation metrics, clearly explain why the RNS is more power efficient than the TCS for addition and multiplication.

1) Switching capacitance in RNS units is lower than TCS

As an example, we illustrate the case of the 32-bit MADD unit when the TCS and RNS implementations have roughly the same area in different corners (DSE for multiply-add in the Supplemental Material):

MADD	corner	area $[\mu m^2]$	power $[\mu W]$
TCS	*ma*	1715 (+1%)	370 (+45%)
RNS	*Ms*	1695	203

The total node count is 909 for TCS and 960 for RNS, and the average capacitance per internal node are C_{ave}(TCS) $= 4.0\ fF$ and C_{ave}(RNS) $= 3.8\ fF$.

The histogram in Fig. 10 (top) shows the capacitances of the circuit internal nodes (obtained as cumulative sum) in different intervals of activity. For example, in the interval $0.4 < a_i \leq 0.6$, the total capacitance switched is about 2 pF for the RNS implementation and about 1.2 pF for the TCS one. Next to the bars, in the figure, we list the node count

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

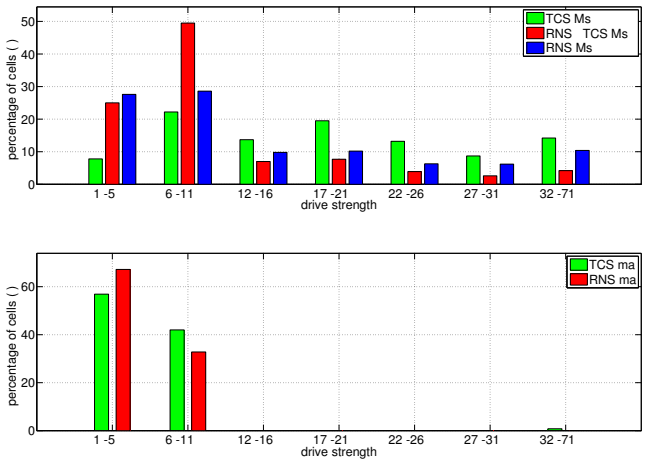

Fig. 11: Cells' drive strength histogram for 32-bit TCS and RNS MADD in *Ms* (top) and *ma* (bottom) corners.

in each interval of activity (e.g., 419 nodes for RNS and 152 nodes for TCS in the interval $0.4 < a_i \leq 0.6$).

The histogram of the dynamic power dissipation (Fig. 10 bottom) is obtained by the cumulative sum of the product $C_{Li} \cdot a_i$ for nodes belonging to each interval.

The experimental results show that the TCS unit has a high number of internal nodes switching at high activity rates, compared to the input activity of 0.5: about 45% the total power in the TCS unit is generated by nodes switching at $a > 1.5$ (bars in the right hand side of Fig. 10), while for the RNS the contribution is just 5% of the total power.

In other words, for two units of about the same area, most of the nodes in the RNS unit switch at lower activity rates resulting in a lower power consumption. Moreover, a large portion of the total power dissipation in the TCS MADD is due to the high switching activity.

2) RNS units have fewer high drive cells

In Fig. 11, we show the histogram of the drive capability of the standard cells mapped by the synthesizer for the 32-bit MADD. The histograms show the percentages of cells of the same drive capability (indicated by the labels "1X", "2X", etc.) in the circuit. Clearly, lower drive capability cells are smaller and dissipate less power.

In the *Ms* corner, Fig. 11 (top), two RNS units are reported: one synthesized with the same time constraint of the TCS *Ms* corner (central red bars in the figure), and one synthesized for the RNS *Ms* corner – faster than the TCS *Ms* corner – (blue bars at right in the figure).

The result of Fig. 11 show that in the *Ms* corner the RNS MADD is synthesized with fewer high-drive cells than the TCS MADD, even when synthesized at RNS maximum speed (faster than TCS *Ms* implementation).

In the opposite design corner (i.e., minimum area) both MADD units are synthesized mostly with low drive cells, and again, in the RNS implementation the percentage of low drive cells is slightly higher than in the TCS one.

3) RNS implementations generate less glitches

As previously mentioned, the main reason for using RNS in digital systems is the reduced carry chain (delays) when a large dynamic range is parallelized in several channels.

Power due to glitches (%) - (TCS max-speed)									
dyn (bit)	16	20	24	28	32	36	40	44	48
TCS	43%	40%	45%	44%	45%	49%	46%	47%	49%
RNS	25%	27%	28%	29%	34%	31%	35%	34%	35%

TABLE 1: Percentage of power due to glitches in MADDs synthesized at TCS max-speed.

Shorter paths have shorter delays because, in general, the signals *'travel'* through a fewer gates (circuit logical depth), and consequently, the spurious transitions due to delay mismatches are reduced.

By defining a *"glitchy"* node, a node with switching activity $a_i > 1.0$ (the clock switching activity is $a_C = 1$), in Fig. 10 bars at the right of "activity = 1" represent glitchy nodes. Therefore, Fig. 10 shows that the TCS MADD is more affected by glitches than the RNS MADD.

To confirm the different impact of glitches on power dissipation, we extracted the glitch count, as explained in Section 3.2.3, for MADDs of different dynamic-range synthesized at the maximum speed achievable by the TCS MADD.

The results, displayed in Table 1, confirm that the RNS units generate fewer glitches than TCS units.

5 Design Methodology

In this section, we explain how our DSE-based methodology can be used to design energy efficient RNS filters. The core of this approach is the DSE performed on the RNS operators.

We sketch in Fig. 12 the proposed tool-chain and design steps. Once the characterization is done, for each RNS modulus, we store the results for delay, area, and power dissipation as a function the timing constraint in *"views"* (a tabular format) of a database with search key m (modulus). Then, the search of the optimal RNS base covering the given dynamic range at the given constraints is done by *BaseFinder*, a tool derived from [28].

BaseFinder evaluates all the combinations of moduli covering the specified dynamic range and meeting the required timing constraint. Then, depending on the target of the implementation (minimum area or lowest power dissipation), the combinations are sorted in increasing cost. *BaseFinder* outputs the lowest cost RNS base in a specification file that is used by *VHDL Builder* [28] to generate the RTL-level code of the processor and the scripts necessary for simulation, synthesis and power estimation.

The main advantage of this approach is that the DSE and the tools take care of all the detail of implementing a RNS system, including conversions, greatly simplifying the tasks of the designer.

The key point is the use of small prime moduli that can be aggregated easily to form the base, and that in such prime moduli, multiplication can be implemented by isomorphism since the indices tables are small.

Next, we show that this design approach is quite accurate and provides reliable solutions.

As example, we consider a programmable 64-tap FIR filter in transposed form with a 24-bit dynamic range. We assume the TCS implementation of this filter as the reference design.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

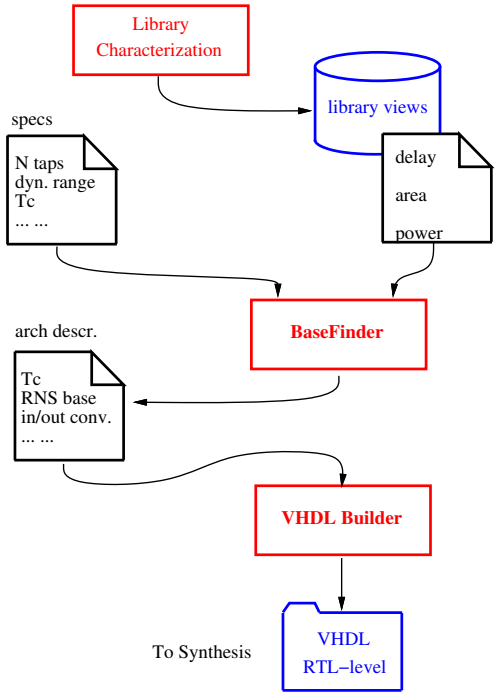

Fig. 12: DSE-based methodology flow to design RNS filters.

#	mod. set comment	RNS base m_1	m_2	m_3	m_4	m_5	m_6	m_7	b	OH	$\lceil \log_2 M \rceil$
1	best	5	7	11	13	17		256	27	3	25
2	2nd best	5	7	13	17	19		128	27	3	25
3	median	5	7	11	13	3		128	26	2	25
4	10% best	7	11	13	17	31		32	26	2	25
5	20% best	5	7	11	17	19	23	8	28	4	25

TABLE 2: Selected RNS bases after DSE for $T_C = 1.2\ ns$ and $d = 24$.

The 64-tap TCS filter is synthesized in the STM 45 nm standard cell library used in the DSE, and its maximum operating frequency (*Ms* corner) is about 830 MHz, corresponding to a clock period of 1.2 ns.

We use this time constraint $T_C = 1.2\ ns$ to run *BaseFinder* for the 64-tap RNS filter covering a 24-bit dynamic range. *BaseFinder* evaluated that more than 24,000 combinations of moduli satisfying the timing constraint can cover the 24-bit dynamic range. By setting as cost criteria the minimum power dissipation, we selected the five RNS bases reported in Table 2.

In Table 2, set 1 is the RNS base with the best power efficiency (lowest cost in the DSE). Set 2, is the second best. Set 4 is the closest moduli set which has cost +10% with respect to the best (set 1). Set 3 is the one which cost is the median value between the costs of sets 1 and 4. This cost may be different from the +5% cost of the best, since it is chosen based on the median position in the sorted ranking of the costs. Set 5 is the RNS base which has cost +20% with respect to set 1. The combinations of moduli which fall in the lowest 20% of the cost are 65. Table 2 also reports the number of bits b necessary to represent the RNS base, its coding overhead OH, and the equivalent dynamic range covered by the base: $\lceil \log_2 M \rceil \geq d$.

The RNS filters of sets 1–5 are generated by *VHDL Builder* and synthesized by Synopsys Design Compiler. The RNS filters are completed with input and output converters, pipelined (by *VHDL Builder*) to match the timing constraint $T_C = 1.2\ ns$. The power estimation is done by running a simulation of the filter programmed as a low-pass filter and white noise (random values) as input.

Filter impl.		Area $[\mu m^2]$	ratio	$P_{ave}^{(1)}$ $[mW]$	ratio$^{(2)}$	ratio$^{(3)}$
TCS		192,584	1.000	80.475	1.000	-
RNS						
1	best	111,481	0.579	44.565	0.554	1.000
2	2nd best	106,962	0.555	46.078	0.573	1.031
3	median	103,920	0.540	46.094	0.573	1.034
4	10% best	107,450	0.558	45.938	0.571	1.034
5	20% best	117,781	0.612	52.650	0.654	1.181

$^{(1)}$ P_{ave} at 200 MHz; $^{(2)}$ ratio to TCS; $^{(3)}$ ratio to RNS set 1.

TABLE 3: 64-tap 24-bit filter comparison: reference TCS vs. RNS (selected RNS bases).

The implementation results for the TCS reference filter and the RNS filters are reported in Table 3. The average power dissipation is evaluated at a clock rate of 500 MHz (T_C=2 ns) for all filters. The two "ratio" columns refer to the P_{ave} ratio to the TCS and RNS set 1, respectively.

Since we are operating in the *Ms* corner and the dynamic range is relatively large, the RNS filters are quite more power efficient than the TCS filter. However, with this experiment, we want to highlight that the power efficiency is quite independent of the chosen RNS base as long as the selection is done based on the DSE. Since the value for the power in Set 5 is about 18% larger than Set 1 – we were expecting 20% from the DSE – we drop Set 5 in the remaining discussion.

The four sets 1–4 are selected among 12 combinations corresponding to the DSE costs within 10% of the lowest cost. The standard deviation for the measured P_{ave} in the filters of sets 1–4 in Table 3 is 0.74 mW corresponding to a variability of about 5% (Fig. 13). Relative to the power dissipation of the TCS filter, this variability has a negligible impact when computing the power efficiency of TCS vs. RNS filters (less than 1%).

In other words, we can trust the DSE to choose the moduli set to be optimal for the given constraints within a confidence of 95%.

Therefore, by using this DSE based approach, we can design power efficient RNS filters without spending time in the trade-off analysis of different RNS bases because the optimal solution is quite insensitive to the specific chosen set. The characterization based DSE and *BaseFinder* take care of the trade-off analysis for the specific design constraints, and *VHDL Builder* generates the RTL code implementing the selected architecture.

To confirm the results of the DSE in Section 3, we implemented the 64-tap 24-bit dynamic range FIR filter in the minimum area corner. The minimum timing constraint resulting in minimum area for the TCS filter is $T_C = 4.0\ ns$.

By running *BaseFinder* with T_C=4.0 ns, the lowest power dissipation RNS base is obtained for the moduli set $\{5, 7, 11, 13, 31, 128\}$, with b=26 and OH=2.

The power estimation, done for the same low pass filter

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2020.2997067, IEEE Transactions on Emerging Topics in Computing

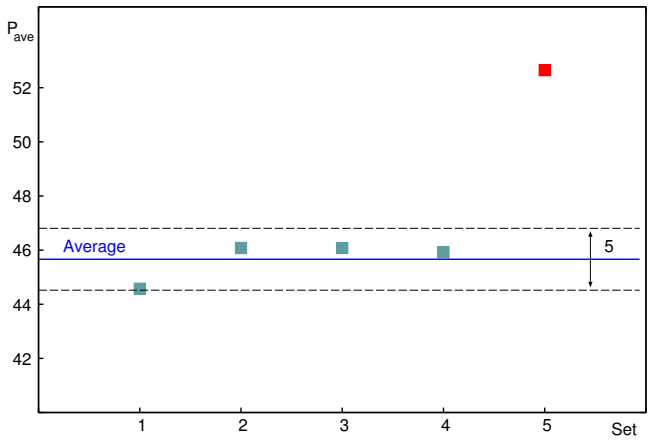

Fig. 13: Plot of P_{ave} for RNS filters. Average and confidence interval are for sets 1–4.

and input patterns, resulted in the following power dissipation (at 250 MHz)

$$\text{TCS: } P_{ave} = 16.82\ mW, \qquad \text{RNS: } P_{ave} = 18.07\ mW$$

By removing the power dissipated in the registers of the FIR filters, the figures for the combinational part are

$$\text{TCS: } P_{ave}^{comb} = 12.46\ mW, \qquad \text{RNS: } P_{ave}^{comb} = 11.98\ mW$$

confirming that, the overhead in the registers offsets the power advantages of RNS in the computation. Also the overhead of input/output converters has an higher impact in the *min area* corner.

Moreover, these implementations in the minimum area design corner confirm the experiment in Section 3.6.4, showing a significant reduction of the switching activity in the tails of the TCS filter.

6 Other RNS Filter Implementations

In this section, we report the results of past work supporting the findings of the DSE for digital filters implemented in RNS.

6.1 Real coefficients FIR Filters

In [29], we implemented two real coefficient FIR filters:

- a low dynamic range (16 bits) and low order filter (16 taps/coefficients);
- a high dynamic range (48 bits) and high order filter (64 taps/coefficients);

and compared their implementations is TCS and RNS.

The experimental results on the DSE-based designs of the filters confirmed that for the dynamic ranges typical of FIR filters. the units implemented in RNS are faster than their TCS counterparts, and that for implementations at the same delay constraints, the RNS units are smaller and consume less power (50–70% reduction).

The experiments also confirmed that if the system has to operate at high frequency rates RNS offers the best trade-offs speed/power, while, when the execution rates are not too demanding, implementing the system in RNS does not give significant advantages because of the added complexity (converters and RNS coding overhead).

6.2 Parallel/Serial FIR Filter

In [27], we presented a high-order FIR filter implemented partly in parallel and partly serially. The starting point is a 128-tap FIR filter working at a frequency of 20 MHz and dynamic range d=36 bits. The selected RNS base, by the DSE, is $\{3, 5, 7, 11, 13, 17, 19, 23, 31, 32\}$ (P=10).

Due to the high order and large dynamic range, the filter is "folded" and executed serially on a 16-tap filter working at $8 \times 20 = 160$ MHz. The detail of the implementation is in [27].

The results of the implementation in [27], show that the RNS unit is much larger than the TCS one. The extra area is mostly due to the overhead introduced by RNS in the registers for the filter coefficients and the input samples.

However, with respect to the power dissipation, this overhead is mitigated by the use of clock gating, and the RNS filter results in 12% savings in power dissipation.

6.3 Polyphase Filters

In [30], we presented the design of polyphase filters in Quadratic RNS (QRNS) for filter banks with a large number of channels for systems on-board telecom satellites.

The design of the system is also based on the DSE for a restricted number of moduli used to represent complex numbers in QRNS [4]. The results in [30] are in line with the ones in Section 4: the lower power consumption in the QRNS filter bank, compared to its TCS counterpart, is due to a reduced switching activity in the RNS operators, and a reduced switching capacitance when the activity is similar. That is, in RNS the switching activity is more evenly distributed on the nodes.

7 Conclusions

The comprehensive design exploration demonstrates that for the dynamic ranges typical of DSP applications, the units implemented in RNS are always faster than their TCS counterparts, and that for implementations at the same delay constraints, the combinational part of RNS units is less complex (smaller area and less glitches generated). In contrast, for the storage part (registers) the overhead introduced by the RNS is significant in some cases.

In RNS the switching activity is significantly reduced because multiplications are implemented by isomorphism, and there is less glitching. The reduction in the multipliers' switching activity is less sizeable when one of the operands has reduced dynamic range and is constant for the whole processing, as for coefficients in frequency filters.

The proposed design methodology for RNS filters produces an optimal implementation of the RNS filter targeting given design constraints, and it is quite insensitive to the specific moduli set, if chosen accordingly to the results of the DSE.

Moreover, our tool-chain greatly simplifies the design by hiding the detail of the RNS implementation of arithmetic blocks. However, if the constraints are not too demanding in terms of execution rates, and the number of operation per sample in not high, implementing the system in RNS may not give significant advantages because of the added complexity (converters and representation overhead).

REFERENCES

[1] N. S. Szabo and R. L. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*. McGraw-Hill, 1967.

[2] P. V. Ananda Mohan, *Residue Number Systems: Algorithms and Architectures*. Kluwer Academic Publishers, 2002.

[3] W. Jenkins and B. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 4, pp. 191–201, 1977.

[4] M. Sodestrand, W. Jenkins, G. A. Jullien, and F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. New York: IEEE Press, 1986.

[5] G. C. Cardarilli, A. Nannarelli, and M. Re, "Reducing Power Dissipation in FIR Filters using the Residue Number System," in *Proc. of 43rd IEEE Midwest Symposium on Circuits and Systems*, vol. 1, Aug. 2000, pp. 320–323.

[6] R. Conway and J. Nelson, "Improved RNS FIR Filter Architectures," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, no. 1, pp. 26 – 28, Jan. 2004.

[7] I. Kouretas and V. Paliouras, "Delay-variation-tolerant FIR filter architectures based on the Residue Number System," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2223–2226.

[8] C. Chang, A. S. Molahosseini, A. A. E. Zarandi, and T. F. Tay, "Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications," *IEEE Circuits and Systems Magazine*, vol. 15, no. 4, pp. 26–44, 4th Quarter 2015.

[9] G. C. Cardarilli, S. Pontarelli, M.Re, and A. Salsano, "Totally Fault Tolerant RNS Based FIR Filters," in *Proc. of 14th IEEE International On-Line Testing Symposium*, July 2014.

[10] E. B. Olsen, "RNS Hardware Matrix Multiplier for High Precision Neural Network Acceleration: RNS TPU," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1 – 5.

[11] H. Nakahara and T. Sasao, "A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1 – 5.

[12] S. Salamat, M. Imani, S. Gupta, and T. Rosing, "RNSnet: In-Memory Neural Network Acceleration Using Residue Number System," in *IEEE International Conference on Rebooting Computing (ICRC)*, 2018, pp. 1 – 12.

[13] N. Chervyakov, P. A. Lyakhov, and M. V. Valueva, "Increasing of Convolutional Neural Network Performance using Residue Number System," in *International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2017, pp. 135 – 140.

[14] N. Samimi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Res-DNN: A Residue Number System-Based DNN Accelerator Unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 2, pp. 658 – 671, Feb. 2020.

[15] S. J. Meehan, S. D. O'Neil, and J. J. Vaccaro, "An Universal Input and Output RNS Converter," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 6, pp. 799–803, June 1990.

[16] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Prentice Hall Signal Processing, 2009.

[17] G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re, "Impact of RNS Coding Overhead on FIR Filters Performance," *Proc. of 41st Asilomar Conference on Signals, Systems, and Computers*, pp. 1426–1429, Oct. 2007.

[18] R. Burch, F. Najm, P. Yang, and T. Trick, "A Monte Carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, pp. 63–71, Mar. 1993.

[19] D. Rabe and W. Nebel, "Short circuit power consumption of glitches," in *International Symposium on Low Power Electronics and Design*, 1996, pp. 125–128.

[20] C. Lemmonds and S. Shetti, "A Low Power 16 by 16 Multiplier Using Transition Reduction Circuitry," *Proc. of International Workshop on Low Power Design*, pp. 139–142, 1994.

[21] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.

[22] M. Bayoumi and G. Jullien, "VLSI Implementation of Residue Adders," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 284–288, Mar. 1987.

[23] A. A. Hiasat, "High-speed and reduced-area modular adder structures for RNS," *IEEE Transactions on Computers*, vol. 51, pp. 84–89, Jan. 2002.

[24] S. Galal, O. Shacham, J. S. Brunhaver II, J. Pu, A. Vassiliev, and M. Horowitz, "FPU Generator for Design Space Exploration," in *Proc. 21st IEEE Symposium on Computer Arithmetic (ARITH)*, Apr. 2013, pp. 25–34.

[25] D. Radhakrishnan and Y. Yuan, "Novel Approaches to the Design of VLSI RNS Multipliers," *IEEE Transaction on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, no. 1, pp. 52–57, Jan. 1992.

[26] A. Nannarelli, G. C. Cardarilli, and M. Re, "Power-Delay Tradeoffs in Residue Number System," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, 2003, pp. 413–416.

[27] M. Petricca, P. Albicocco, G. C. Cardarilli, A. Nannarelli, and M. Re, "Power Efficient Design of Parallel/Serial FIR Filters in RNS," *Proc. of 46th Asilomar Conference on Signals, Systems, and Computers*, pp. 1015–1019, Nov. 2012.

[28] A. Del Re, A. Nannarelli, and M. Re, "A Tool for Automatic Generation of RTL-level VHDL Description of RNS FIR Filters," in *Proc. of 2004 Design, Automation and Test in Europe Conference (DATE)*, vol. 48, Feb. 2004, pp. 686–687.

[29] G. C. Cardarilli, A. Nannarelli, M. Petricca, and M. Re, "Characterization of RNS Multiply-Add Units for Power Efficient DSP," in *Proc. of 58th IEEE Midwest Symposium on Circuits and Systems*, Aug. 2015.

[30] G. C. Cardarilli, A. Nannarelli, Y. Oster, M. Petricca, and M. Re, "Design of Large Polyphase Filters in the Quadratic Residue Number System," *Proc. of 44th Asilomar Conference on Signals, Systems, and Computers*, pp. 410–413, Nov. 2010.

Gian Carlo Cardarilli was born in Rome, Italy, and received the "laurea" (summa cum laude) from the University of Rome "La Sapienza," Italy, in 1981. He has been with the University of Rome "Tor Vergata," Italy, since 1984, and he is currently full professor of digital electronics and electronics for communication systems in the Department of Electronic Engineering. During 1992-1994, he was with the University of L'Aquila, Italy. During 1987-1988, he was with the Circuits and Systems team at EPFL, Lausanne, Switzerland. His interests are in the area of VLSI architectures for signal processing and IC design. In this field, he published more than 160 papers in international journals and conferences. He has also regular cooperation with companies such as: Alcatel Alenia Space, Italy; STM, Agrate Brianza, Italy; Micron, Italy; Selex S.I., Italy. His scientific interest concerns the design of special architectures for signal processing. He works in the field of computer arithmetic and its application to the design of fast signal digital processor.

Luca Di Nunzio received the master's degree (summa cum laude) in electronics engineering and the Ph.D. degree in Systems and Technologies for Space Applications from the University of Rome "Tor Vergata," Italy, in 2006 and 2010, respectively. He has a working history with several companies in the fields of electronics and communications. He is currently an Adjunct Professor with the Digital Electronics Laboratory, at the University of Rome "Tor Vergata," Italy, and an Adjunct Professor of digital electronics at University "Guglielmo Marconi," Italy. His research activities are in the fields of reconfigurable computing, communication circuits, digital signal processing, and machine learning.

Rocco Fazzolari received the master's degree in Electronic Engineering in May 2009 and the Ph.D. in Systems and Technologies for Space Applications in June 2013 at University of Rome "Tor Vergata," Italy. At present, he is postdoctoral fellow and assistant professor at University of Rome "Tor Vergata," Italy, in the Department of Electronic Engineering. He works on hardware implementation of high-speed systems for digital signals processing, machine learning and array of wireless sensor networks.

Alberto Nannarelli (S'94-M'99-SM'13) graduated in electrical engineering from the University of Rome "La Sapienza," Italy, in 1988, and received the M.S. and the Ph.D. degrees in electrical and computer engineering from the University of California at Irvine, USA, in 1995 and 1999, respectively. He is an Associate Professor at the Technical University of Denmark, Lyngby, Denmark. He worked for SGS-Thomson Microelectronics and for Ericsson Telecom as a Design Engineer and for Rockwell Semiconductor Systems as a summer intern. From 1999 to 2003, he was with the Department of Electronic Engineering, University of Rome "Tor Vergata, Italy, as a Postdoctoral Researcher. His research interests include computer arithmetic, computer architecture, and VLSI design. Dr. Nannarelli is a Senior Member of the IEEE Computer Society.

Massimo Petricca graduated in electronic engineering from the University of Rome "Tor Vergata," Italy, in 2008, where he also received the Ph.D. degree in Systems and Technologies for Space Applications in 2012. From 2008 to 2012, he worked on computer arithmetic and VLSI designs for low-power digital signal processing, also as a visiting Ph.D. student at the Technical University of Denmark. From 2012 to 2013, he was with the EDA Group at the Polytechnic University of Turin, Italy, as a Postdoctoral Researcher focused on low-power VLSI designs methodologies and CAD automation for battery systems modeling. He joined Elettronica, Rome, Italy, in 2013 as an FPGA Designer, and he is currently a System Engineer in the Research and Innovation department.

Marco Re (M'92) Marco Re holds a Ph.D. in Microelectronics and he is Associate Professor at the University of Rome "Tor Vergata," Italy, in the Department of Electronic Engineering, where he teaches Digital Electronics and Hardware Architectures for DSP. He was awarded with two NATO fellowships at the University of California at Berkeley, USA, working as visiting scientist at the Cadence Berkeley Laboratories. He has been awarded with the Otto Moensted fellowship as visiting professor at the Technical University of Denmark. Marco collaborates in many research projects with different companies in the field of DSP architectures and algorithms. His main scientific interest is in the field of: low power DSP algorithms architectures, hardware-software codesign, fuzzy logic and neural hardware architectures, low power digital implementations based on non-traditional number systems, computer arithmetic and CAD tools for DSP. He is author of about 200 papers in international journals and international conferences. He is member of IEEE and Audio Engineering Society (AES). He is Director of the Master in Audio Engineering at the University of Rome "Tor Vergata," Italy.