VLSI Research Group

University of Windsor

Comments on "An Arithmetic Free Parallel Mixed-Radix Conversion Algorithm"

Submitted to IEEE Trans. Circuits and Systems

Antonio García, *Student Member, IEEE*, and
Graham A. Jullien, *Senior Member, IEEE*

Comments on “An Arithmetic Free Parallel Mixed-Radix Conversion Algorithm”

Antonio García, Student Member, IEEE, and Graham A. Jullien, Senior Member, IEEE

Abstract

In a recently published paper “An Arithmetic Free Parallel Mixed-Radix Conversion Algorithm”, two algorithms based on look-up tables for mixed-radix conversion are presented. Here we show that one of the algorithms had been prior published in 1978, and we also take this opportunity to speak to the use of look-up tables for RNS with present and future technologies.

Keywords: Residue Number System; Look-up Table Arrays; Mixed Radix Conversion

1.0 INTRODUCTION

Look-up table based systems for the development of RNS applications is a well-know solution, but the efficiency of such an approach is very technology dependent. About 20 years ago, one of the authors wrote a paper on the use of ROM Arrays for RNS operations [2]. At that time individually packaged ROMs were an efficient counterpart to LSI arithmetic chips, but in the intervening years, with the advent of VLSI and complete systems on a single chip, binary arithmetic arrays have appeared more efficient than ROM arrays. It was interesting to us, therefore, to read the recently published paper, "An arithmetic free parallel mixed-radix conversion algorithm," by D. F. Miller and W. S. McCormick, in which the idea of only using arrays of look-up tables has again been proposed for RNS implementations. Perhaps technology has come full-circle and small ROMs are now an efficient counterpart to logic gates. Evidence from the SIA Roadmap [3] shows that the exponent for the projected exponential increase in SRAM density is greater than that for packed logic (see Figure 1) and one can therefore probably project the same comparison for ROM density versus packed logic.

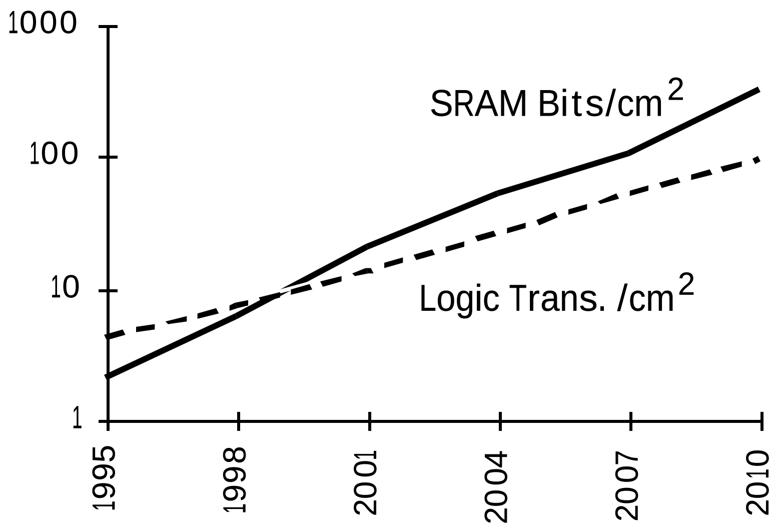

Figure 1. Density projections from the SIA Roadmap

Our own evidence also shows that the ratio of ROM area to the area of arithmetic blocks using logic gates is decreasing as we implement with deeper sub-micron technologies.

Two separate algorithms were published in [1] and in the following section we show that one of the algorithms for MRC (Mixed Radix Conversion) [4] is essentially the same as a conversion algorithm published in [2].

2.0 MATHEMATICAL COMPARISON

We will show in this section that Algorithm I in [1] is introduced as part of the base extension needed for the scaling scheme presented in [2]. Algorithm I assumes a set of moduli $\{m_1, \ldots, m_N\}$ with $1 < m_1 < m_2 < \ldots < m_N$. Thus, each non-negative integer, *x*, has a unique RNS representation $(x_1, \ldots, x_N)$. In the same way, the *mixed radix* representation of *x* is unique and is given by eqn. (1).

$$x = a_1 + a_2 m_1 + a_3 m_2 m_1 + \ldots + a_N m_1 m_2 \ldots m_{N-1} \tag{1}$$

Algorithm I in [1] is simply derived from the fact that $a_1 = x_1$, which can be deduced directly from the definition of the mixed radix representation of *x*. On the other hand, it is clear that the integer $y^{(1)} = \frac{(x - x_1)}{m_1} < m_2 m_3 \ldots m_N$ can be expressed as follows in the RNS defined by $\{m_2, m_3, \ldots, m_N\}$:

$$y^{(1)}_{j+1} = \frac{x - x_1}{m_1} \bmod m_{j+1} = m_{1,\,j+1}(x_{j+1} - x_1) \bmod m_{j+1} \tag{2}$$

where $j = 1, 2, \ldots, N-1$ and $m_{1,\,j+1}$ is the multiplicative inverse of m_1 modulo m_{j+1}. It can be easily shown that $k^{(1)}_j = m_{1,\,j+1}(x_{j+1} - x_1) \bmod m_{j+1} = y^{(1)}_{j+1}$, where $k^{(1)}_j$ is the solution for the Linear Diophantine Equation [1]; in this way, $k^{(1)}_1 = y^{(1)}_2 = a_2$. It can also be deduced from the fact that the mixed radix representation of $y^{(1)}$ is just:

$$y^{(1)} = \frac{x - x_1}{m_1} = a_2 + a_3 m_2 + \ldots + a_N m_2 \ldots m_{N-1} \tag{3}$$

In general, at the end of stage *i*, it is shown inductively that the algorithm outputs $[k^{(i)}_1, k^{(i)}_2, \ldots, k^{(i)}_{N-i}]$, i.e., the RNS representation of eqn. (4):

$$y^{(i)} = \frac{x - a_1 - \sum_{r=2}^{i} a_r m_1 m_2 \dots m_{r-1}}{m_1 m_2 \dots m_i} = a_{i+1} + a_{i+2} m_{i+2} + \dots + a_N m_{i+2} \dots m_{N-1} \quad (4)$$

in the set of moduli $\{m_{i+1}, \dots, m_N\}$; as in previous stages, $k_1^{(i)} = y_{i+1}^{(i)} = a_{i+1}$.

Eqn. (4) corrects an error found in the denominator of equation (10) in reference [1].

We are now going to show that the procedure presented in reference [1] corresponds to the technique described in [2]. Firstly, we must remember that mixed radix conversion appears in the base extension process that is inherent to the data scaling procedure in [2]. If *Y* is the result of scaling *X*, defined in the set of moduli $\{m_0, m_1, \dots, m_{N-1}\}$, by the scaling constant *K*, defined as $K = \prod_{i=0}^{S-1} m_i$ (*S*<*N*), it is obvious that $0 \le Y < \prod_{i=S}^{N-1} m_i$, and so *Y* is completely defined by the subset of moduli $\{m_S, \dots, m_{N-1}\}$. In this way, we can compute the residues $(y_S, \dots, y_{N-1})$ and then perform a base extension process for obtaining $(y_0, \dots, y_{S-1})$. A mixed radix conversion will be part of this process of base extension, thus we can obtain the mixed radix representation of *Y* from $(y_S, \dots, y_{N-1})$, and then perform a mixed radix to RNS conversion to obtain $(y_0, \dots, y_{S-1})$.

It is not our intention to describe here the complete scaling and base extension process, so we are only going to recall the fundamental equations for mixed radix conversion in [2] (for clarity we will use the notation found in [2]). Thus, we can now define the mixed radix representation of *Y* as follows:

$$Y = r_0 + \sum_{i=1}^{N-S-1} r_i \prod_{j=0}^{i-1} m_{j+S} \quad (5)$$

since $0 \le Y < \prod_{i=S}^{N-1} m_i$. If we define $r_j = \left|R^{(j)}\right|_{m_{j+S}}$, we can solve eqn. (5) with the recursive equation:

$$\left|R^{(k+1)}\right|_{m_i} = \left|\left|R^{(k)} - r_k\right|_{m_i} \cdot \left|\frac{1}{m_{k+S}}\right|_{m_i}\right|_{m_i} \tag{6}$$

where $\left|\frac{1}{m_{k+S}}\right|_{m_i}$ is defined as the multiplicative inverse of m_{k+S} modulo m_i ($m_{k+S,\,i}$ using the above notation) and $R^{(0)} = Y$.

We now show that the recursive procedure in eqn. (6) is the same algorithm as described in reference [1]; this is immediate, since it is clear that $R^{(k)}$ in eqn. (6) is equivalent to $y^{(k)}$ in eqn. (4), taking account the differences in the indexing of the equations. In this way, the algorithm described in reference [2] starts with the RNS representation of the data to be mixed radix converted, i.e., Y in this case, and the first mixed radix digit is simply $r_0 = \left|R^{(0)}\right|_{m_S} = |Y|_{m_S} = y_S$, in the same way that a_1=x_1 in Algorithm I of reference [1].

At this point, let us show the expressions for r_1 and r_2:

$$r_1 = \left|R^{(1)}\right|_{m_{S+1}} = \left|\left|R^{(0)} - r_0\right|_{m_{S+1}} \cdot \left|\frac{1}{m_S}\right|_{m_{S+1}}\right|_{m_{S+1}} = \left|\frac{R^{(0)} - r_0}{m_S}\right|_{m_{S+1}} \tag{7}$$

$$\begin{aligned} r_2 &= \left|R^{(2)}\right|_{m_{S+2}} = \left|\left|R^{(1)} - r_1\right|_{m_{S+2}} \cdot \left|\frac{1}{m_{S+1}}\right|_{m_{S+2}}\right|_{m_{S+2}} = \left|\frac{R^{(1)} - r_1}{m_{S+1}}\right|_{m_{S+2}} \\ &= \left|\frac{R^{(0)} - r_0 - r_1 m_S}{m_S m_{S+1}}\right|_{m_{S+2}} \end{aligned} \tag{8}$$

these can be easily extended, by inductive reasoning, for the higher index mixed radix digits. Thus it is clear that eqn. (7) and (8) reproduce the same computation procedure that is described in reference [1] (eqn. (3) and (4)). Thus, the mixed radix conversion proposed in [2] requires the computation of the residues of $R^{(k)}$ with respect to the set of moduli $\{m_{S+k}, \ldots, m_{N-1}\}$ during the k-th stage; this computation requires, for each modulus, the correspondent residue $\left|R^{(k-1)}\right|_{m_i}$ $(S + k \le i \le N - 1)$ and the previously computed mixed radix digit $r_{k-1} = \left|R^{(k-1)}\right|_{m_{S+k-1}}$ in order to calculate eqn. (6). In this way, residues $\left|R^{(k)}\right|_{m_i}$ $(S + k + 1 \le i \le N - 1)$ are passed to the next stage, and a new mixed radix digit $r_k = \left|R^{(k)}\right|_{m_{S+k}}$ is obtained. This is obviously, the same procedure as described in [1], and the structure shown in Figures 3 and 4 (with T_3 functions) in [2] for mixed radix conversion is completely analogous to that shown in Figures 1 and 2 in [1]. In this way, the subtraction and multiplication by a constant factor (the corresponding multiplicative inverse) performed by these two algorithms through eqn. (4) and eqn. (6), respectively, can be stored in a double input look-up table, so the complete mixed radix conversion process can be performed entirely with tables. As a final note, in addition to a table-based mixed radix conversion technique, two table-based scaling algorithms were also presented in reference [2].

3.0 REFERENCES

[1] D. F. Miller and W. S. McCormick, “An arithmetic free parallel mixed-radix conversion algorithm,” *IEEE Trans. Circuits Syst. II Analog and Digital Signal Processing*, vol. 45, pp. 158-162, Jan. 1998.

[2] G. A. Jullien, “Residue number scaling and other operations using ROM arrays,” *IEEE Trans. Comput.*, vol. C-27, pp. 325-336, Apr. 1978.

[3] “The National Technology Roadmap for Semiconductors”, Semiconductor Industry Association, 1994.

[4] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology.* New York: McGraw-Hill, 1967.

4.0 Affiliation of authors:

Antonio García is with the Departamento de Electrónica y Tecnología de Computadores, University of Granada, 18071 Granada, Spain. He is supported by the Dirección General de Enseñanza Superior under project PB96-1397.

Graham A. Jullien is with the VLSI Research Group, University of Windsor, Ontario, Canada N9B 3P4.