Information Processing Letters 115 (2015) 453–457

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A note on RNS architectures for the implementation of the diagonal function

Stanisław J. Piestrak [1]

Institut Jean Lamour (UMR 7198 CNRS)/Université de Lorraine, Res. Team 406 MAE, Faculté des Sciences et Technologies, BP 70239, F-54506 Vandœuvre Lès Nancy, France

ARTICLE INFO

Article history:
Received 10 July 2009
Received in revised form 26 November 2014
Accepted 2 December 2014
Available online 9 December 2014
Communicated by A. Tarlecki

Keywords:
Computer architecture
Parallel processing
Residue number system

ABSTRACT

This paper focuses on some considerations on the diagonal function and its applications to implement non-modular operations like magnitude comparison and sign detection in residue number system (RNS), recently proposed in the literature. According to our results, any implementation involving the diagonal function proposed to date results in excessive hardware overhead and delay, which make it impractical from the application point of view, so that it cannot compete with more traditional approaches.

1. Introduction

The residue number system (RNS) is a non-positional number system whose inherently parallel nature enables very high data throughput to be obtained owing to very fast virtually carry-free arithmetic [1]. Three basic arithmetic operations (add, subtract, and multiply) are easily implemented in RNS and performed on operands significantly shorter than the dynamic range of RNS. On the other hand, the difficulties with implementations of some non-modular operations in RNS (such as division, magnitude comparison, sign detection, and overflow detection) have been the reasons for limited applications of the RNS.

Recall that an RNS is defined by a set of n positive integers $\{m_1, m_2, \ldots, m_n\}$ which are pairwise relatively prime. The *dynamic range* M of the RNS with n moduli, i.e., the number of different integers that can be uniquely represented in the RNS, is given by $M = \prod_{i=1}^{n} m_i$. In an RNS, a numerical value of a natural number X in the range $[0, M-1]$ is represented by an n-tuple $\{x_1, x_2, \ldots, x_n\}$, whose components are the residues of X with respect to an ordered set of moduli m_i, where $x_i = X \bmod m_i = |X|_{m_i}$, $0 \le |X|_{m_i} \le m_i - 1$, and $i = 1, 2, \ldots, n$. The number of bits needed to represent residues mod m_i and M will be denoted $a_i = \lceil \log_2 m_i \rceil$ and $a = \lceil \log_2 M \rceil$, respectively.

Conceptually the most obvious approaches to execute non-modular operations in RNS are based on mixed radix conversion (MRC) and the Chinese remainder theorem (CRT) [1]. However, the MRC requires $n(n-1)/2$ look-up tables for its implementation and $n-1$ cycles for its execution, whereas the CRT requires an n-operand adder modulo a large number M with inputs provided e.g. by n look-up tables. Some drawbacks of the methods relying on these approaches have motivated researchers to consider some alternative solutions to implement non-modular RNS operations in hardware.

As for the magnitude comparison of two numbers $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ $(0 \le X, Y \le M-1)$, which is of our interest here, the most obvious approach relies on the residue-to-binary conversion of X and Y followed by the comparison of the positional representations of X and Y. Throughout the years, several new mathematical concepts have been proposed in attempt to facilitate

E-mail address: stanislaw.piestrak@univ-lorraine.fr.

[1] Part of this research was done when the author was on leave with IRISA/INRIA, Res. Team CAIRN, 6, rue de Kérampont, 22300 Lannion, France.

http://dx.doi.org/10.1016/j.ipl.2014.12.003
0020-0190/

execution of difficult operations in RNS, and in particular magnitude comparison [2–6]. One group of approaches relies on using the 'core function' introduced in [2]. According to [2], in most cases it suffices to compare the values of the core functions for the two numbers but, unfortunately, for some 'critical core' values, the magnitude comparison requires a further time-consuming iterative process. Then, Miller et al. [3] have attempted to resolve the latter problem by using a redundant modulus which, however, not only reduces the useful dynamic range of the RNS but also introduces significant hardware overhead. Finally, Gonnella [4] has suggested an alternative definition of the core function and introduced the so-called 'skin function' which measures the non-linearity of the core function. As a result, non-modular operations in RNS (like magnitude comparison) can be executed: (i) in parallel for non-'critical-core' regions (provided that the dynamic range is slightly restricted to avoid critical cores); and (ii) iteratively for 'critical-core' regions of the dynamic range of the RNS (which is a sequential process however). The other group of approaches relies on using the 'diagonal function' considered in [5,6]. The basic architecture of the magnitude comparator of [5] was generalized in [6], where new RNS architectures for the effective implementation of the diagonal function were proposed. The superiority of the new architectures for magnitude comparison was claimed with respect to traditional approaches in terms of hardware amount and time delay.

The purpose of this paper is to comment on a few aspects of various implementations of the diagonal function and their usefulness for magnitude comparison in the RNS. The first one is that most delay assumptions regarding basic circuits are underestimated as a matter of fact. The second one is that the hardware complexity of some blocks has not been taken into account. It means that all presented delay and hardware complexity evaluations are misleading and that the trivial magnitude comparator based on any converter from RNS to the positional system followed by a comparator remains the best solution. Finally, some hidden drawbacks of the sum of quotients technique (SQT) used both in [5] and [6] are revealed.

2. Magnitude comparison using diagonal function

The 'diagonal function' is a theoretical concept introduced in [5] and extended in [6] to improve performance of magnitude comparison and other non-modular operations in the RNS. It is defined as:

$$\mathbf{D}(X) = \left| \sum_{i=1}^{n} k_i \cdot x_i \right|_{SQ}, \tag{1}$$

where: the sum of quotients $SQ = \sum_{i=1}^{n} M/m_i$ is called the 'diagonal modulus' of the RNS; and $k_i = |-1/m_i|_{SQ}$, $i = 1, 2, \ldots, n$ ($|1/m_i|_{SQ}$ is the multiplicative inverse of m_i modulo SQ, i.e. $|k_i \cdot m_i|_{SQ} = 1$). The basic magnitude comparison algorithm of X and Y using diagonal function, presented in [5], relies on the following properties of the diagonal function:

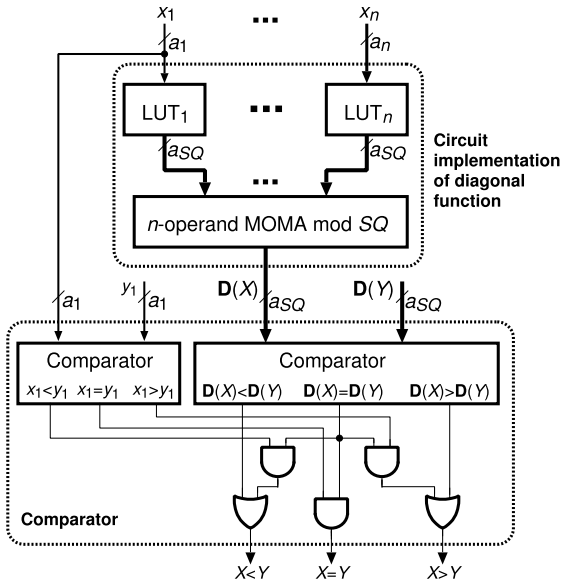

Fig. 1. Implementation of the basic magnitude comparison algorithm using diagonal function of [5].

1. $\mathbf{D}(X) < \mathbf{D}(Y) \Rightarrow X < Y$;
2. $\mathbf{D}(X) > \mathbf{D}(Y) \Rightarrow X > Y$;
3. $\mathbf{D}(X) = \mathbf{D}(Y) \Rightarrow \begin{cases} x_i < y_i \Rightarrow X < Y; \\ x_i > y_i \Rightarrow X > Y; \\ x_i = y_i \Rightarrow X = Y. \end{cases}$

To reduce delay, we suggest the hardware implementation of the basic magnitude comparison algorithm using diagonal function of [5], as shown in Fig. 1. For the case 3, we explicitly recommend to select for comparison the smallest modulus m_i with the minimum length $\lceil \log_2 m_i \rceil$ of operands; henceforth, without loss of generality, it is assumed that $m_1 = \min_{1 \le i \le n}\{m_i\}$. Compared to delay estimations provided in [6], the scheme of Fig. 1 entails two modifications: (i) because two comparators of $\mathbf{D}(X)$ vs. $\mathbf{D}(Y)$ and x_1 vs. y_1 can operate separately and in parallel, it suffices to take into account only the delay of the former; (ii) two extra gate levels of the final AND-OR gates must be included. Obviously, the circuit that implements the diagonal function is used twice for X and Y to produce $\mathbf{D}(X)$ and $\mathbf{D}(Y)$, prior the comparison $\mathbf{D}(X)$ vs. $\mathbf{D}(Y)$ can take place (similar principle applies to all other methods considered here).

3. Delay assumptions

Here, we shall follow the same notation as already used in [5,6]. The basic blocks used for magnitude comparison in RNS are: $MOMA(n, a)$ — a multi-operand modular adder (MOMA) for n operands with a-bit word length (with a time delay denoted $t_{MOMA(n,a)}$); $L(l, a)$ — a look-up table of 2^l locations with a-bit output word length (with a time delay denoted $t_{L(l,a)}$); $C(a)$ — a binary comparator with a-bit word length (with a time delay denoted $t_{C(a)}$). In all complexity evaluations the same MOMA from [7], used also in [6], will be used here. It is built using the tree of carry-save adders (CSAs) followed by the