Handling zero in diminished-one modulo $2^n + 1$ adders

CONSTANTINOS EFSTATHIOU[†], HARIDIMOS T. VERGOS*[‡] and DIMITRIS NIKOLOS[‡]

Zero treatment in diminished-one modulo $2^n + 1$ addition has traditionally been performed separately, leading to slow and area-consuming implementations. To overcome this, on the basis of an enhanced number representation used previously, we introduce novel carry look ahead and parallel-prefix architectures for diminished-one modulo $2^n + 1$ adders that can also handle operands equal to 0. Translators for the new representation are also given.

1. Introduction

Modulo $2^n + 1$ arithmetic appears to play an important role in many applications. The Residue Number System (RNS) (Sonderstrand *et al.* 1986, Bayoumi *et al.* 1987, Elleithy and Bayoumi 1992, Koren 2002) is one of the first application fields. A set of L moduli, say $\langle m_1, m_2, \ldots, m_L \rangle$, that are pair-wise relative prime, is used to define an RNS. Any integer X, with $0 \leq X < M$, where $M = m_1 \times m_2 \times \cdots \times m_L$, has a unique representation in the RNS, given by the L-tuple of residues $X = (x_1, x_2, \ldots, x_L)$, where $x_i = X \bmod m_i$. An RNS operation, say $*$, is defined as $(z_1, z_2, \ldots, z_L) = (x_1, x_2, \ldots, x_L) * (y_1, y_2, \ldots, y_L)$, where $z_i = (x_i * y_i) \bmod m_i$. In most RNS applications $*$ is addition, subtraction or multiplication. Since the computation of z_i depends only upon x_i, y_i and m_i, each z_i is computed in parallel in a separate arithmetic unit, often called a channel. Moduli choices of the form $\langle 2^n, 2^n - 1, 2^n + 1 \rangle$ (Jenkins and Leon 1977) have received significant attention because they offer very efficient circuits in the area $\times$ time^2 product sense (Paliouras and Stouraitis 1999). Addition in such systems is performed using three channels that in fact are a modulo $2^n - 1$ (equivalently one's complement), a modulo 2^n and a modulo $2^n + 1$ adder (Sonderstrand *et al.* 1986, Koren 2002).

Modulo $2^n + 1$ adders are also utilized as the last-stage adders of modulo $2^n + 1$ multipliers. Modulo $2^n + 1$ multipliers find applicability in pseudorandom number generation (Lehmer 1951), cryptography (Lai and Massey 1990, Curiger 1993, Zimmermann *et al.* 1994) and in the Fermat number transform, which is an effective way to compute convolutions (Ma 1998).

The addition delay in an RNS application which uses the $\langle 2^n, 2^n - 1, 2^n + 1 \rangle$ moduli is dictated by the modulo $2^n + 1$ channel, since this requires $(n + 1)$-bit wide operands. To overcome the problem of dealing with $(n + 1)$-bit wide circuits

Received 10 February 2002. Accepted 15 May 2003.

*Corresponding author. e-mail: vergos@ceid.upatras.gr

†Department of Informatics, TEI of Athens, Ag. Spyridonos Street, 12210 Egaleo, Athens, Greece.

‡Department of Computer Engineering and Informatics, University of Patras, 26500, Patras, Greece, and Computer Technology Institute, 3 Kolokotroni Street, 26221 Patras, Greece.

International Journal of Electronics ISSN 0020–7217 print/ISSN 1362–3060 online © 2003 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/0020721031000148263

for the modulo $2^n + 1$ channel, the diminished-one number system (Leibowitz 1976) has been adopted in several residue number system implementations, for example Jenkins (1979, 1982). In this system each number X is represented by $X^* = X - 1$, with $X^* \in [0, 2^n - 1]$. Therefore the adoption of the diminished-one number system leads to modulo $2^n + 1$ adders and multipliers of n bits wide operands (Zimmermann 1997, 1999, Efstathiou *et al.* 2001, Vergos *et al.* 2001, 2002). Implementations of diminished-one adders that are as fast as the modulo 2^n and modulo $2^n - 1$ ones have been reported in Vergos *et al.* (2001, 2002). However, a new problem arises: the special treatment required for operands equal to 0. The previous proposals on diminished-one adders (Zimmermann 1997, 1999, Efstathiou *et al.* 2001, Vergos *et al.* 2001, 2002) totally ignore this subject with the exception of Agrawal and Rao (1978). To overcome the problem of zero operands, Agrawal and Rao (1978) proposed to adopt in their representation a zero indication bit and showed how Carry Look Ahead (CLA) adders that can handle zero operands can be implemented. CLA adders that require one or two cycles have been proposed in Agrawal and Rao (1978). However, the single cycle CLA adders proposed in Agrawal and Rao (1978) do not handle correctly either the modulo $2^n + 1$ carries or the zero indication bit of the result (Efstathiou *et al.* 2002).

In this paper, we adopt a number representation similar to that of Agrawal and Rao (1978). We present three architectures for modulo $2^n + 1$ addition, that can also handle zero operands. Our CLA architecture requires a single addition cycle and has been derived by engaging the output carry equation into the CLA unit. Our parallel-prefix adders with a fast carry increment stage have the same logical depth and hence speed as well as the same area complexity as the diminished-one adders with a carry increment stage proposed in Zimmermann (1997, 1999) that cannot handle zero operands. To further decrease the number of required prefix levels, we propose a totally parallel-prefix architecture that cancels the need for a separate carry increment stage by performing carry recirculation at each prefix level (Kalamboukas *et al.* 2000, Vergos *et al.* 2001, 2002). Our totally parallel-prefix adders offer the same logical depth and complexity as the fastest modulo 2^n (Kogge and Stone 1973, Ladner and Fischer 1980), modulo $2^n - 1$ (Kalamboukas *et al.* 2000) and diminished-one adders (Vergos *et al.* 2001, 2002), with the extra advantage of handling zero operands. Converters between the adopted representation and the modulo $2^n + 1$ binary number representation are finally given.

2. Problem definition

Suppose that a number X, with $0 \le X \le 2^n$, is represented by $n+1$ bits, as $x_z X^* = x_z x^*_{n-1} x^*_{n-2} \ldots x^*_1 x^*_0$, where x_z is the zero indication bit and X^* is the diminished-one representation of X; that is:

$$x_z = \begin{cases} 0, & \text{if } X \neq 0 \\ 1, & \text{if } X = 0 \end{cases} \quad \text{and} \quad X^* = \begin{cases} X - 1, & \text{if } X \neq 0 \\ 0, & \text{if } X = 0 \end{cases}$$

Obviously, $X = X^* + \overline{x_z}$ (the overbar notation is used for logical negation). A similar representation was adopted in Agrawal and Rao (1978). If we decide to ignore the representation of 0 and concentrate only on the X^* part, then efficient n-bits wide adders can be constructed as proposed in Zimmermann (1997, 1999), Efstathiou *et al.* (2001) and Vergos *et al.* (2001, 2002). However, the need to treat 0 distinctly will

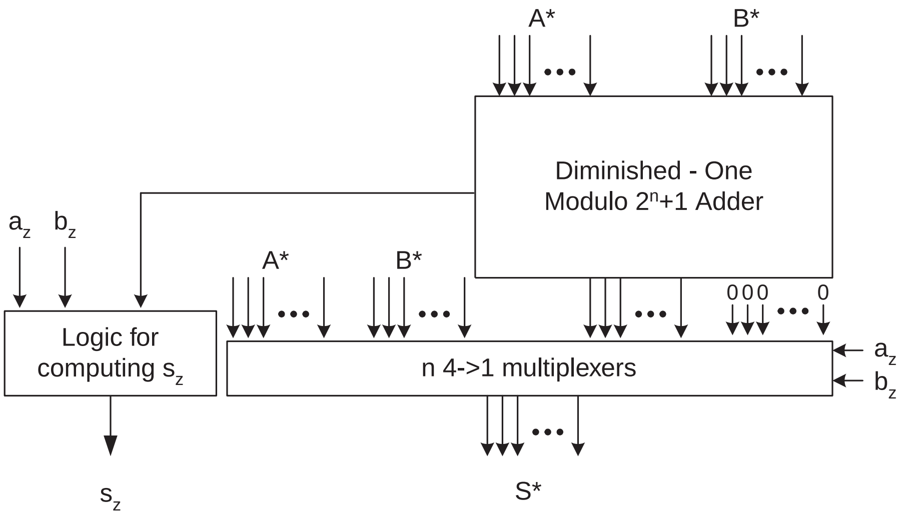

Figure 1. The architecture of an adder that treats 0 as a special case.

result in a circuit like the one shown in figure 1. At the output of the adder a row of multiplexers is added in order to select the correct output. When none of the operands is zero (both a_z and b_z are 0), the output of the adder is propagated. When one of the two operands is zero, the other operand is allowed to propagate. When both operands are 0, the all-0s word is propagated. Logic also needs to be added for computing the zero indication bit s_z of the result.

Obviously, the adders that follow the architecture of figure 1 are not the best choice in RNS applications, since the delay of the multiplexers makes the delay of the modulo $2^n + 1$ channel greater than that of channels of the form 2^n and $2^n - 1$. The required implementation area is also increased substantially in comparison with the other channels.

3. The new proposed modulo $2^n + 1$ adders

In this section we propose CLA and parallel-prefix adders with embedded treatment of zero operands. We first analyse the logic of s_z and S^*, for the adopted number representation. Let $|X|_Y$ denote the modulo Y residue of X.

Relation $|A + B|_{2^n+1} = 0$, with $0 \le A, B \le 2^n$, implies that $A = B = 0$ or that $A, B \neq 0$ and $A + B = 2^n + 1$, or equivalently that $A = B = 0$, or $A, B \neq 0$ and $A^* + B^* = 2^n - 1$. Therefore, the zero indication bit, s_z, of the result should be one when either both operands are 0, or when both operands are non-zero but their diminished-one parts are complementary. These two cases are expressed by the following relation for s_z:

$$s_z = (a_z \cdot b_z) \vee \left(\overline{a_z} \cdot \overline{b_z} \cdot P_{n-1}\right) = (a_z \cdot b_z) \vee \left(\overline{(a_z \vee b_z)} \cdot P_{n-1}\right) \tag{1}$$

where $P_{n-1} = p_{n-1} \cdot p_{n-2} \cdot \cdots \cdot p_0$ and $p_i = a_i^* \oplus b_i^*$, with $i = 0,1, \ldots, n-1$, and the notations $\cdot, \vee, \oplus$ are used for the logical AND, OR and exclusive-OR operations respectively. An implementation of (1) is presented in figure 2. In some adders (often referred as exclusive-OR adders) the carry propagate signals are defined as

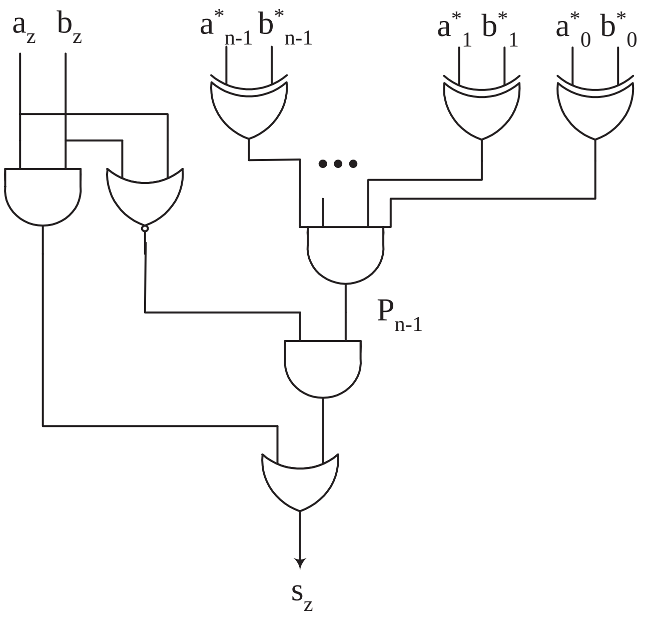

Figure 2. The implementation of s_z.

the exclusive-OR of corresponding operands' bits. In these adders, the logic required for producing P_{n-1} is already present in the adder. On the other hand, the logic producing P_{n-1} is required in inclusive-OR adders, that is, in adders whose carry propagate signals are defined as the inclusive-OR of corresponding operands' bits.

For the S^* part of the result we can observe that:

- If A^* and B^* are both non-zero, then according to Zimmermann (1997, 1999):

$$S^* \bmod (2^n + 1) = \begin{cases} (A^* + B^*) \bmod 2^n, & \text{if } A^* + B^* \geq 2^n \\ A^* + B^* + 1 & \text{otherwise} \end{cases}$$

  or, equivalently, in this case S^*can be computed by adding the complement of the carry output ($\overline{c_{\text{out}}}$) of the modulo 2^n addition of A^* and B^* back to the sum. Since in this case $a_z = b_z = 0$, we can, instead of $\overline{c_{\text{out}}}$, add $\overline{(a_z \vee b_z \vee c_{\text{out}})}$.
- If one or both operands are zero, $\overline{(a_z \vee b_z \vee c_{\text{out}})} = 0$ and S^* is equal to the modulo 2^n sum of A^* and B^*. That is, in this case S^* can also be computed by a modulo 2^n adder with a carry input of $\overline{(a_z \vee b_z \vee c_{\text{out}})}$.

According to the above analysis, the computation of the diminished-one part of the result, S^*, can in all cases, be carried out by a modulo 2^n adder whose carry input is connected to a circuit implementing the expression $F = \overline{(a_z \vee b_z \vee c_{\text{out}})}$. However, such a direct connection will lead to an oscillating and thus slow circuit. One straightforward way of avoiding oscillations is to compute two possible sums in two distinct modulo 2^n adders and then use function F for choosing between the two sums. However, since this arrangement requires two sets of n-bit adders and n multiplexers, it increases the required implementation area significantly. In the following we propose CLA and parallel-prefix architectures that do not suffer from oscillations.

3.1. *CLA adders*

Let $g_k = a_k^* \cdot b_k^*$ and $p_k = a_k^* \oplus b_k^*$ denote the carry generate and propagate terms respectively. The carries c_k, with $-1 \leq k \leq n-1$ (c_{-1} is the input carry, while $c_{n\text{-}1}$ the output carry) in an n-bit CLA adder are computed by unfolding the recursive equation $c_k = g_k \vee p_k \cdot c_{k-1}$ and, in parallel, implementing the resulting equations. For the inverted carry at each bit position c_k, we have

$$\overline{c_k} = \overline{g_k \vee p_k \cdot c_{k-1}} = \overline{g_k} \cdot (\overline{p_k} \vee \overline{c_{k-1}}) = \overline{g_k} \cdot \overline{p_k} \vee \overline{g_k} \cdot \overline{c_{k-1}} = \overline{t_k} \vee \overline{g_k} \cdot \overline{c_{k-1}} \qquad (2)$$

where $t_k = \overline{\overline{g_k} \cdot \overline{p_k}} = a_k^* \vee b_k^*$. The sum bits are given by $s_k = p_k \oplus c_{k-1}$, for $0 \leq k \leq n-1$.

Dedicated CLA architectures that do not suffer from oscillations are derived in the following by engaging $F = \overline{(a_z \vee b_z \vee c_{\text{out}})}$ in (2) and simplifying the resulting equations (Agrawal and Rao 1978, Efstathiou *et al.* 1994, Vergos *et al.* 2002). This procedure will provide us with the equations of the carries for the modulo 2^n+1 addition, hereafter denoted by c_i^*, with $-1 \leq i \leq n-2$. Using (2), for the least significant carry, we get

$$\begin{aligned} c_{-1}^* &= F = \overline{a_z \vee b_z \vee c_{n-1}} = \left(\overline{a_z \vee b_z}\right) \cdot \overline{c_{n-1}} = \left(\overline{a_z \vee b_z}\right) \cdot (\overline{t_{n-1}} \vee \overline{g_{n-1}} \cdot \overline{c_{n-2}}) \\ &= \left(\overline{a_z \vee b_z}\right) \cdot (\overline{t_{n-1}} \vee \overline{g_{n-1}} \cdot (\overline{t_{n-2}} \vee \overline{g_{n-2}} \cdot \overline{c_{n-3}})) \\ &= \left(\overline{a_z \vee b_z}\right) \cdot (\overline{t_{n-1}} \vee \overline{g_{n-1}} \cdot \overline{t_{n-2}} \vee \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \overline{c_{n-3}}) = \cdots \\ &= \left(\overline{a_z \vee b_z \vee t_{n-1}}\right) \vee \left(\overline{a_z \vee b_z \vee g_{n-1}}\right) \cdot \overline{t_{n-2}} \vee \left(\overline{a_z \vee b_z \vee g_{n-1}}\right) \\ &\quad \cdot \overline{g_{n-2}} \cdot \overline{t_{n-3}} \vee \cdots \vee \left(\overline{a_z \vee b_z \vee g_{n-1}}\right) \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1} \cdot \overline{g_0} \end{aligned} \qquad (3)$$

Using (3) and the recursive equation $c_k = g_k \vee p_k \cdot c_{k-1}$ we can then derive the rest of the carries' equations. For $k=0$ and by substituting c_{-1}^* by (3), we get

$$\begin{aligned} c_0^* &= g_0 \vee p_0 \cdot c_{-1}^* \\ &= g_0 \vee p_0 \cdot \left(\left(\overline{a_z \vee b_z \vee t_{n-1}}\right) \vee \left(\overline{a_z \vee b_z \vee g_{n-1}}\right)\right. \\ &\quad \left. \cdot \overline{t_{n-2}} \vee \cdots \vee \left(\overline{a_z \vee b_z \vee g_{n-1}}\right) \cdot \overline{g_{n-2}} \cdots \overline{g_2} \cdot \overline{g_1} \cdot \overline{g_0}\right) \end{aligned} \qquad (4)$$

Given that $g_0 \vee p_0 \cdot \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1} \cdot \overline{g_0} = g_0 \vee p_0 \cdot \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1}$ and that $p_0 \cdot \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1} \vee p_0 \cdot \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{t_1} = p_0 \cdot \overline{g_{n-1}} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1}$, (4) becomes

$$\begin{aligned} c_0^* &= g_0 \vee p_0 \cdot \overline{(a_z \vee b_z \vee t_{n-1})} \vee p_0 \cdot \overline{(a_z \vee b_z \vee g_{n-1})} \cdot \overline{t_{n-2}} \vee \\ &\quad \cdots \vee p_0 \cdot \overline{(a_z \vee b_z \vee g_{n-1})} \cdot \overline{g_{n-2}} \cdot \cdots \cdot \overline{g_2} \cdot \overline{g_1} \end{aligned}$$

We then can use c_0^* for computing c_1^*, c_1^* for computing c_2^* and so on, up to

$$\begin{aligned} c_{n-2}^* &= g_{n-2} \vee p_{n-2} \cdot g_{n-1} \vee \cdots \vee p_{n-2} \cdot p_{n-3} \cdot \cdots \cdot p_1 \cdot g_0 \\ &\quad \vee p_{n-2} \cdot p_{n-3} \cdot \cdots \cdot p_1 \cdot p_0 \cdot \overline{(a_z \vee b_z \vee g_{n-1})} \end{aligned}$$

The above equations sum up to the following general formula, which defines our proposed one-level CLA diminished-one modulo $2^n + 1$ architecture:

$$c_i^* = g^*_{|n+i|_n} \vee \sum_{j=0}^{n-2} \left(\prod_{k=j+1}^{n-1} p^*_{|k+i+1|_n} \right) g^*_{|j+i+1|_n}$$

where

$$p_j^* = \begin{cases} \overline{g_j}, & \text{if } n-1 > j > i+1 \\ \overline{a_z \vee b_z \vee g_j}, & \text{if } j = n-1 \\ p_j, & \text{otherwise} \end{cases} \quad \text{and} \quad g_j^* = \begin{cases} \overline{t_j}, & \text{if } n-1 > j > i+1 \\ \overline{a_z \vee b_z \vee p_j}, & \text{if } j = n-1 \\ \overline{g_j}, & \text{if } j = i+1 \\ g_j, & \text{otherwise} \end{cases} \tag{5}$$

3.2. *Parallel-prefix adders with carry increment stage*

To solve the problem of oscillations in parallel-prefix adder architectures, a first solution is to use prefix architectures with fast carry processing as proposed in Abraham and Gajski (1980) and then utilize the theory developed in Zimmermann (1997, 1999), for re-entering the $\overline{(a_z \vee b_z \vee c_{\text{out}})}$ expression as the carry input. The resulting architecture is outlined in figure 3. Figure 4 presents the gate-level implementations of the operators used in figure 3. The prefix computation unit can be designed using any of the proposed prefix algorithms (Kogge and Stone 1973, Ladner and Fischer 1980, Brent and Kung 1982, Knowles 2001, Beaumont-Smith and Lim 2001). The notations G_i and P_i are used in figure 3 to denote group generate

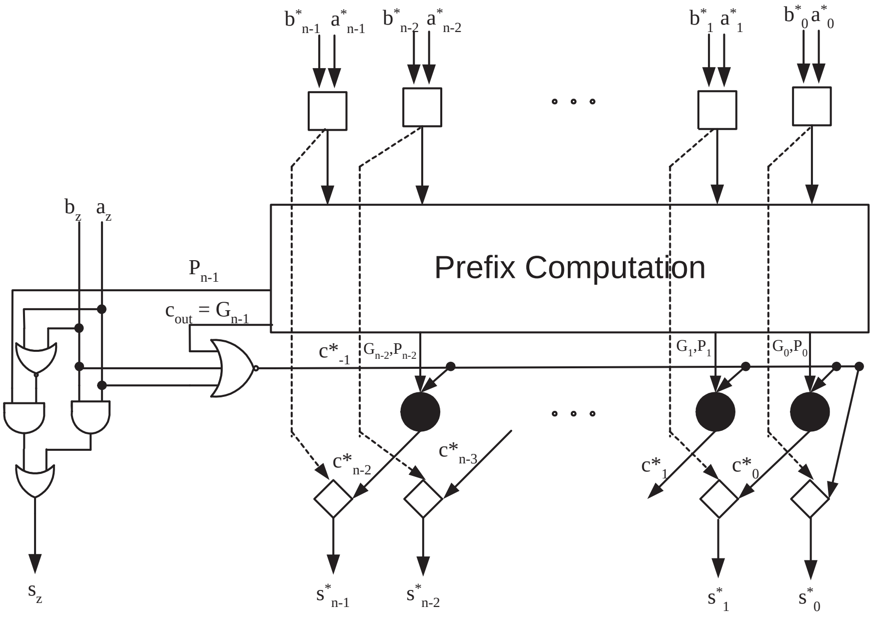

Figure 3. The architecture of a parallel-prefix adder with a carry increment stage.

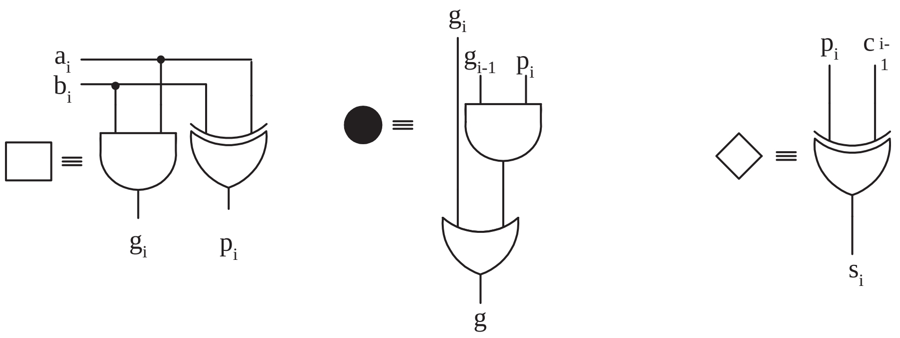

Figure 4. Gate-level implementations of the operators used in figure 3.

and group propagate signals of bits 0, 1, …, i. That means that G_i and P_i are respectively the first and the second members of the group relation (assuming that carry input $c_{\text{in}} = 0$):

$$(G_i, P_i) = \begin{cases} (g_0, p_0), & \text{if } i = 0 \\ (g_i, p_i) \circ (G_{i-1}, P_{i-1}), & \text{if } 1 \le i \le n-1 \end{cases}$$

and the operator $\circ$ is defined according to Brent and Kung (1982) as $(g_m, p_m) \circ (g_k, p_k) = (g_m \vee p_m \cdot g_k,\ p_m \cdot p_k)$. In an integer adder, obviously the carry at position i is $c_i = G_i$.

Comparing the architecture of figure 3 against the parallel-prefix architecture with a carry increment stage proposed in Zimmermann (1997, 1999), for diminished-one modulo $2^n + 1$ addition it is obvious that both architectures have similar area and time complexities. However, the architecture of figure 3 in parallel offers treatment of zero operands.

The number of prefix levels required by the architecture of figure 3 is one more than those of the fastest modulo 2^n (Kogge and Stone 1973, Ladner and Fischer 1980), modulo $2^n + 1$ (Kalamboukas *et al.* 2000) and modulo $2^n + 1$ diminished-one adder architectures (Vergos *et al.* 2001, 2002), because the latter do not require a carry increment stage. Therefore, in an RNS application that uses the $\langle 2^n, 2^n - 1, 2^n + 1 \rangle$ moduli set, the architecture of figure 3 limits the maximum execution speed of the system.

3.3. *Totally parallel-prefix adders*

Instead of having a dedicated single stage for the re-entering carry, it has been proposed in Kalamboukas *et al.* (2000) and Vergos *et al.* (2001, 2002) to perform carry recirculation at each existing prefix level. In this way the need for an extra carry increment stage is cancelled, and dedicated totally parallel-prefix adder architectures result with one less prefix level.

In the case that the re-entering carry is given by the expression $\overline{(a_z \vee b_z \vee c_{\text{out}})}$, allowing carry recirculation at each existing prefix level, we can ascertain that the carries c_i^* of the modulo $2^n + 1$ addition are equal to G_i^*, where G_i^* is computed by the prefix equations

$$(G_i^*, P_i^*) = \begin{cases} \overline{(G_{n-1}, P_{n-1})}, & \text{if } i = -1 \\ (G_i, P_i) \circ \overline{(G_{n-1,i+1}, P_{n-1,i+1})} & \text{if } 0 \leq i \leq n-2 \end{cases} \tag{6}$$

where

- $\overline{(G, P)}$ is defined to be equal to $(\overline{G}, P)$;
- $G_{a,b}$ and $P_{a,b}$, $(a > b)$, are respectively the group generate and propagate signals for the group a, $a-1, \dots, b$, computed by $(G_{a,b}, P_{a,b}) = (g_a, p_a) \circ (g_{a-1}, p_{a-1}) \circ \cdots \circ (g_b, p_b)$. Obviously, $(G_{a,0}, P_{a,0}) = (G_a, P_a)$, and
- $g_{n-1} = (a_{n-1}^* \cdot b_{n-1}^*) \vee a_z \vee b_z$.

In several cases, the equations indicated by (6) require more than $\log_2 n$ prefix levels for their implementation. As shown in Vergos *et al.* (2001, 2002), these equations can be transformed into equivalent ones that can all be implemented within $\log_2 n$ prefix levels. For the sake of completeness, we present the main idea of the theory in Vergos *et al.* (2001, 2002) below.

If $(G_x, P_x) = (g, p) \circ \overline{(G, P)}$ and $(G_y, P_y) = \overline{((\overline{t}, \overline{g}) \circ (G, P))}$, then, since $G_x = g \vee p \cdot \overline{G} = \overline{(g \vee p \cdot \overline{G})} = \overline{(\overline{g} \cdot (\overline{p} \vee G))} = \overline{(\overline{g} \cdot \overline{p} \vee \overline{g}G)} = \overline{(\overline{t} \vee \overline{g}G)}$ and $G_y = \overline{(\overline{t} \vee \overline{g}G)}$, we get $G_x = G_y$. This implies that a carry whose equation is of the form $(g, p) \circ \overline{(G, P)}$ can be equivalently computed by an equation of the form $\overline{((\overline{t}, \overline{g}) \circ (G, P))}$. As shown in Vergos *et al.* (2002) for area-time efficient parallel-prefix modulo $2^n + 1$ adder implementations the above transformation should be applied j times recursively to the equations of the form $(g_i, p_i) \circ \cdots \circ (g_0, p_0) \circ \overline{(G_{n-1,i+1}, P_{n-1,i+1})}$ produced by (6), until:

$$n - 1 - i + j = \begin{cases} n, & \text{if } i > \frac{n}{2} - 1 \\ \frac{n}{2}, & \text{if } i \leq \frac{n}{2} - 1 \end{cases}$$

Example. For the implementation of a modulo 17 adder that can also handle zero operands, from (6) we get the following set of prefix equations :

$$\begin{aligned} c_{-1}^* &= \overline{((g_3, p_3) \circ (g_2, p_2) \circ (g_1, p_1) \circ (g_0, p_0))} \\ c_0^* &= (g_0, p_0) \circ \overline{((g_3, p_3) \circ (g_2, p_2) \circ (g_1, p_1))} \\ c_1^* &= (g_1, p_1) \circ (g_0, p_0) \circ \overline{((g_3, p_3) \circ (g_2, p_2))} \\ c_2^* &= (g_2, p_2) \circ (g_1, p_1) \circ (g_0, p_0) \circ \overline{(g_3, p_3)}, \end{aligned}$$

where $g_3 = a_z \vee b_z \vee (a_3^* \cdot b_3^*)$. We then transform the above into the following equations that can all be implemented within a prefix tree with a logical depth of 2:

$$\begin{aligned} c_{-1}^* &= \overline{((g_3, p_3) \circ (g_2, p_2) \circ (g_1, p_1) \circ (g_0, p_0))} \\ c_0^* &= \overline{((\overline{t_0}, \overline{g_0}) \circ (g_3, p_3) \circ (g_2, p_2) \circ (g_1, p_1))} \\ c_1^* &= (g_1, p_1) \circ (g_0, p_0) \circ \overline{((g_3, p_3) \circ (g_2, p_2))} \\ c_2^* &= (g_2, p_2) \circ (g_1, p_1) \circ \overline{(\overline{t_0}, \overline{g_0}) \circ (g_3, p_3)}. \end{aligned}$$

Figure 5 presents the attained implementation along with explanation of newly introduced operators.

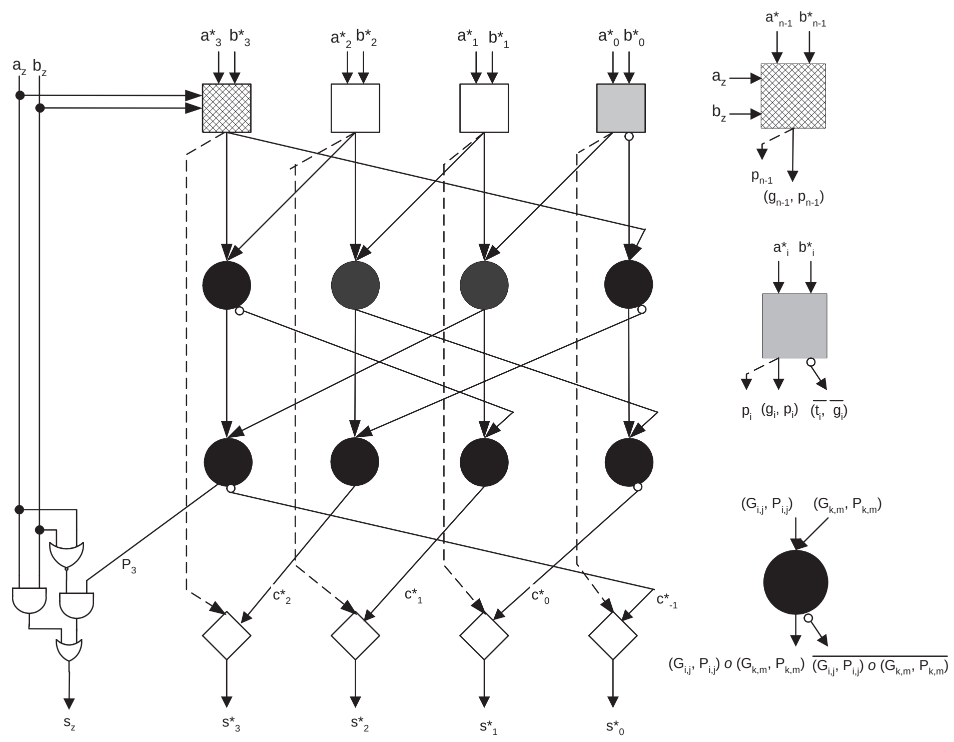

Figure 5. Proposed parallel-prefix adder with minimal logical depth.

The number of prefix levels and therefore the delay of the proposed totally parallel-prefix adders is equal to that of the fastest reported modulo 2^n (Kogge and Stone 1973, Ladner and Fischer 1980), $2^n - 1$ (Kalamboukas *et al.* 2000) and $2^n + 1$ diminished-one (Vergos *et al.* 2001, 2002) adders. This makes them highly applicable in RNS applications. Moreover, their hardware (gates and routing) complexity is analogous to that of the adders reported in Kalamboukas *et al.* (2000) and Vergos *et al.* (2001, 2002).

4. Translator circuits

4.1. *Translator from modulo $2^n + 1$ to the proposed representation*

Let $X = x_n x_{n-1} x_{n-2} \cdots x_1 x_0$ be a binary number with $0 \le X \le 2^n$ and $x_z X^* = x_z x^*_{n-1} x^*_{n-2} \cdots x^*_1 x^*_0$ the targeted representation. The zero indication bit x_z can be computed by

$$x_z = \overline{x_n \vee x_{n-1} \vee \ldots \vee x_1 \vee x_0} \tag{7}$$

while

$$X^* = \begin{cases} X - 1, & \text{if } x_z = 0 \\ 0, & \text{if } x_z = 1 \end{cases}$$

or, equivalently,

$$X^* = X - 1 + x_z = \left|X + 2^n - 1 + x_z\right|_{2^n}.$$

The last relation reveals that X^* can be computed by a modulo 2^n adder that accepts as inputs the all-1s operand and the n least significant bits of X operand and, as carry input the x_z signal. Assuming an inclusive-OR implementation of the adder, we have that $g_i = x_i$ and $t_i = 1$. Therefore, utilizing (7) we obtain that the carry at each position i is given by

$$\begin{aligned} c_i &= g_i \vee t_i \cdot c_{i-1} = x_i \vee c_{i-1} = \cdots = x_i \vee x_{i-1} \vee \cdots \vee x_0 \vee c_{\text{in}} \\ &= x_i \vee x_{i-1} \vee \cdots \vee x_0 \vee \overline{(x_n \vee x_{n-1} \vee \cdots \vee x_1 \vee x_0)} \end{aligned}$$

The latter relation reveals that the adder required for implementing a translator from the binary system to the adopted representation is composed of an exclusive-NOR gate per bit and of a carry computation unit easily implemented as trees of NOR gates.

4.2. *Translator from the adopted representation to binary*

Using the notation, introduced, above in this case we have that $X = X^* + \overline{x_z}$. The latter relation reveals that translation to binary can be performed by a simple incrementer.

5. Conclusions

Several architectures have recently been proposed for diminished-one modulo $2^n + 1$ addition. None of these has dealt with the problem of handling zero operands. All of them propose to treat zero operands separately. Unfortunately, such a treatment leads to slow and area consuming implementations.

In this paper, by utilizing a number representation similar to that of Agrawal and Rao (1978), we proposed CLA and parallel-prefix adders able also to handle zero operands. Our CLA adders perform addition in a single cycle and were derived by engaging the equation of the re-entering carry into the carry computation equations. The parallel-prefix adders proposed herein, with a carry increment stage offer the same logical depth and hence speed as well as the same area complexity as the diminished-one adders with a carry increment stage proposed in Zimmermann (1997, 1999), with the extra advantage of handling zero operands. The proposed totally parallel-prefix adders perform carry recirculation at each prefix level and therefore do not need a separate carry increment stage; their number of prefix levels and therefore their execution speed is the same as that of the fastest modulo 2^n, modulo $2^n - 1$ and modulo $2^n + 1$ diminished-one adders. Translators between the adopted number representation and the modulo $2^n + 1$ binary system were finally presented.

Acknowledgements

This research was partially supported by the Research Committee of Patras University within the framework of the K. Karatheodoris scholarships programme.

References

Abraham, J. A., and Gajski, D. D., 1980, Easily testable high-speed realization of register-transfer-level operations. *Proceedings of the 10th Fault-Tolerant Computing Symposium (FTCS-10)*, Kyoto, Japan, pp. 339–344.

Agrawal, D. P. and Rao, T. R. N., 1978, Modulo ($2^n + 1$) arithmetic logic. *Electronic Circuits and Systems*, **2**, 186–188.

Bayoumi, M. A., Jullien, G. A., and Miller W. C., 1987, A look-up table VLSI design methodology for RNS structures used in DSP applications. *IEEE Transactions on Circuits and Systems*, **CAS-34**, 604–616.

Beaumont-Smith, A., and Lim, C. C., 2001, Parallel prefix adder design. *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, Los Alamitos, CA, USA, pp. 218–225.

Brent, R. P., and Kung, H. T., 1982, A regular layout for parallel adders. *IEEE Transactions on Computers*, **C-31**, 260–264.

Curiger, A., 1993, VLSI architectures for computations in finite rings and fields. PhD thesis, Swiss Federal Institute of Technology.

Efstathiou, C., Nikolos, D., and Kalamatianos, J., 1994, Area–time efficient modulo $2^n - 1$ adder design. *IEEE Transactions on Circuits and Systems—II*, **41**, 463–467.

Efstathiou, C., Vergos, H. T. and Nikolos, D., 2001, On the design of modulo $2^n \pm 1$ adders. *Proceedings of the 8th IEEE International Conference on Electronics, Circuits & Systems* (*ICECS 2001*), Malta, vol. I, pp. 517–520.

Efstathiou, C., Vergos, H. T., and Nikolos, D., 2002, Handling zero in diminished-one modulo $2^n + 1$ adders. Technical Report No. 2002-01-03, Computer Technology Institute, Patras, Greece.

Elleithy, K. M., and Bayoumi, M. A., 1992, Fast and flexible architectures for RNS arithmetic decoding. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, **CAS-39**, 226–235.

Jenkins, W. K., 1979, Recent advance in residue number techniques for recursive digital filtering. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-27**, 19–30.

Jenkins, W. K., 1982, The design of specialized residue classes for efficient recursive digital filter realization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-30**, 370–380.

Jenkins, W. K., and Leon, B. J., 1977, The use of residue number systems in the design of finite impulse response digital filters. *IEEE Transactions on Circuits and Systems*, **CSA-24**, 191–201.

Kalamboukas, L., Nikolos, D., Efstathiou, C., Vergos, H. T., and Kalamatianos, J., 2000, High-speed parallel-prefix modulo $2^n - 1$ adders. *IEEE Transactions on Computers, Special Issue on Computer Arithmetic*, **49**, 673–680.

Knowles, S., 2001, A family of adders. *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, Los Alamitos, CA, USA, pp. 277–284.

Kogge, P. M., and Stone, H. S., 1973, A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, **22**, 783–791.

Koren, I., 2002, *Computer Arithmetic Algorithms*, 2nd edn (Nattick, MA: A. K. Peters Publications).

Ladner, R. E., and Fischer, M. J., 1980, Parallel prefix computation. *Journal of the ACM*, **27**, 831–838.

Lai, X., and Massey, J. L., 1990, A proposal for a new block encryption standard. *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, Aarhus, Denmark, *Lecture Notes in Computer Science 473*, 1999, pp. 389–404.

Lehmer, D. H., 1951, *Proceedings of the 2nd Symposium on Large-scale Digital Calculating Machinery* (Cambridge, MA: Harvard University Press), pp. 141–146.

Leibowitz, L. M., 1976, A simplified binary arithmetic for the Fermat number transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-24**, 356–359.

Ma, Y., 1998, A simplified architecture for modulo ($2^n + 1$) multiplication. *IEEE Transactions on Computers*, **47**, 333–337.

Paliouras, V., and Stouraitis, T., 1999, Novel high-radix residue number system multipliers and adders. *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'99)*, Orlando, FL, USA, pp. 451–454.

Sonderstrand, M. A., Jenkins, W. K., Jullien, G. A., and Taylor, F. J., 1986, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing* (New York: IEEE Press).

Vergos, H. T., Efstathiou, C., and Nikolos, D., 2001, High speed parallel-prefix modulo $2^n + 1$ adders for diminished-one operands. *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, Vail, CO, pp. 211–217.

Vergos, H. T., Efstathiou, C., and Nikolos, D., 2002, Diminished-one modulo $2^n + 1$ adder design. *IEEE Transactions on Computers*, **51**, 1389–1399.

Zimmermann, R., 1997, Binary adder architectures for cell-based VLSI and their synthesis. PhD thesis, Swiss Federal Institute of Technology.

Zimmermann, R., 1999, Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication. *Proceedings of the 14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, Adelaide, Australia, pp. 158–167.

Zimmermann, R., Curiger, A., Bonnenberg, H., Kaeslin, H., Felber, N., and Fichtner, W., 1994, A 177 Mb/s VLSI implementation of the international data encryption algorithm. *IEEE Journal of Solid-State Circuits*, **29**, 303–307.