2009 19th IEEE International Symposium on Computer Arithmetic

Unified Approach to the Design of Modulo-($2^n \pm 1$) Adders Based on Signed-LSB Representation of Residues

Ghassem Jaberipur [1]
Dept. Electrical & Computer Engr.
Shahid Beheshti Univ., Tehran, Iran
jaberipur@sbu.ac.ir

Behrooz Parhami
Dept. Electrical & Computer Engr.
Univ. of California, Santa Barbara, USA
parhami@ece.ucsb.edu

Abstract

Moduli of the form $2^n \pm 1$, which greatly simplify certain arithmetic operations in residue number systems (RNS), have been of longstanding interest. A steady stream of designs for modulo-($2^n \pm 1$) adders has rendered the latency of such adders quite competitive with ordinary adders. The next logical step is to approach the problem in a unified and systematic manner that does not require each design to be taken up from scratch and to undergo the error-prone and labor-intensive optimization for high speed and low power dissipation. Accordingly, we devise a new redundant representation of mod-($2^n \pm 1$) residues that allows ordinary fast adders and a small amount of peripheral logic to be used for mod-($2^n \pm 1$) addition. Advantages of the building-block approach include shorter design time, easier exploration of the design space (area/speed/power tradeoffs), and greater confidence in the correctness of the resulting circuits. Advantages of the unified design include the possibility of fault-tolerant and gracefully degrading RNS circuit realizations with fairly low hardware redundancy.

1. Introduction

Residue number systems (RNS), which have been studied since the early days of digital computers, have again come into the forefront [6] in view of improved algorithms for some of the difficult operations and the desire to use plentiful transistors for performance improvement without an undue energy burden. Moduli of the form $2^n \pm 1$, which greatly simplify certain RNS arithmetic operations have also been of longstanding interest. Over the years, many articles have been published on the design of modulo-($2^n \pm 1$) adders (e.g., [1], [2], [5]) and other arithmetic circuits, with new proposals still appearing on a regular basis [7]. This progression has gradually reduced the latency of such designs to the point of being quite competitive with ordinary (mod-2^n) adders.

[1] G. Jaberipur is also affiliated with School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

The next logical step is to approach the problem in a unified and systematic manner that does not require designs to be taken up from scratch and to undergo the error-prone and labor-intensive optimization for speed and low power dissipation with each implementation. We present a design method that constitutes a first step in this direction. More specifically, we devise a new redundant representation of mod-($2^n \pm 1$) residues that enables ordinary fast adders, and a small amount of extra logic, to realize mod-($2^n \pm 1$) addition for any n.

We show that designs based on our *signed-LSB representation* are faster and/or less complex than existing ones, in all but one case where some speed is lost. The latter case does not lead to any performance degradation, because the corresponding adder is off the critical path in an RNS processor. Furthermore, the fact that our designs can be based on conventional fast adders, carry-save adders, and other standard arithmetic building blocks allows for a shorter design time, easier exploration of the design space in terms of area/speed/power tradeoffs, greater confidence in the correctness of the resulting circuit implementations, and simpler testing. Any improvement in, or new tradeoff strategy for, such building blocks that occurs from time to time, would yield corresponding benefits for our modular adders, with no extra effort.

The rest of this paper is organized as follows. After discussing existing designs for modulo-($2^n \pm 1$) adders in Section 2, we introduce the signed-LSB redundant representation in Section 3 and use it for designing our modular adders in Section 4. The practicality of our approach hinges upon simple conversions from/to binary representation, a topic covered in Section 5. In Section 6, cost-performance comparisons, based on both gate-level analysis and circuit synthesis, are presented, along with a discussion of the applicability of our approach to the design of fault-tolerant RNS processors. Cost and speed comparisons are performed with a corrected version of a previously published design [1]; refer to our on-line supplement [4] for a description of the flaw in the original design, along with the corrected mod-($2^n + 1$) adder.

1063-6889/09 $25.00 © 2009 IEEE
DOI 10.1109/ARITH.2009.14

2. Modulo-($2^n \pm 1$) Adders

Modulo-m addition of mod-m residues A and B ($0 \le A, B < m$) is defined as:

$$S = |A + B|_m = \begin{cases} A + B - m, & \text{if } A + B \ge m \\ A + B, & \text{otherwise} \end{cases} \quad (1)$$

Replacing m in Eqn. 1 with $2^n - 1$ or $2^n + 1$ yields the corresponding equations for mod-(2^n–1) or mod-(2^n+1) addition, respectively. Because comparing $A + B$ with $2^n - 1$ or $2^n + 1$ is nontrivial, Eqn. 1 is modified into Eqns. 2 and 3, which use much simpler comparisons with 2^n. Here, $W = (w_n w_{n-1} \ldots w_1 w_0)_{\text{two}} = A + B$ is the true sum of A and B, which can be decomposed into a single bit w_n and an n-bit number $|W|_{2^n}$. Similarly, $W' = (w'_n w'_{n-1} \ldots w'_1 w'_0)_{\text{two}} = A + B - 1$ is the *diminished sum* of A and B, with its associated decomposition into a single bit w'_n and an n-bit number $|W'|_{2^n}$.

$$S^- = |A+B|_{2^n-1} = \left.\begin{cases} W - 2^n + 1, & \text{if } W \ge 2^n \\ W, & \text{otherwise} \end{cases}\right\} = |W|_{2^n} + w_n \quad (2)$$

$$S^+ = |A+B|_{2^n+1} = \left.\begin{cases} W' - 2^n, & \text{if } W' \ge 2^n \\ W' + 1, & \text{otherwise} \end{cases}\right\} = |W'|_{2^n} + \overline{w'}_n \quad (3)$$

Note that the term $\overline{w'}_n = 1 - w'_n$ in Eqn. 3 is the logical complement of w'_n. Equation 2 always yields the correct modulo-($2^n - 1$) sum, except when $W = 2^n - 1$, in which case it produces $S^- = 2^n - 1$, an alternate representation of $|0|_{2^n-1}$. Hardware realization of Eqn. 2 (3) entails the use of end-around carry (inverted carry) in the addition process that normally produces W (W').

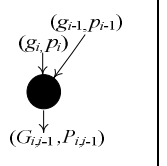
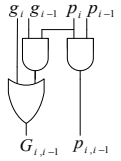
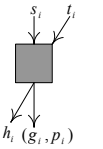
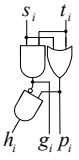
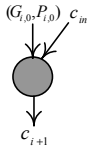
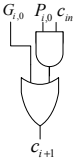
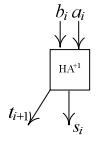
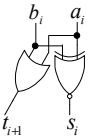
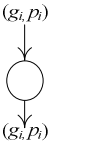
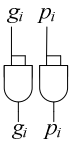
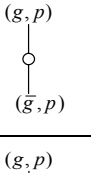
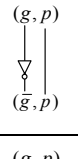
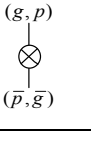
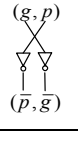
Kalamboukas *et al.* [5] have implemented Eqn. 2, to compute $|A + B|_{2^n-1}$, via a totally parallel prefix (TPP) adder. This TPP design, depicted in Fig. 1b for $n = 8$, has a latency of 2 log n + 3 unit gate delays (UGD), where UGD has been defined in [12]. Each black circle in Fig. 1b represents the carry operator (see Table 1), and there are n crossing lines for partial end-around carry signals. The alternate design shown in Fig. 1a, based on a less complex regular parallel prefix (RPP) adder, imposes an extra carry operation level, enclosed in a dashed box, for accommodating the end-around carry. This leads to overall latency of 2 log n + 5 UGD.

Computation of $W' = A + B - 1$, required by Eqn. 3, is not as simple as $W = A + B$. However, there is a way out of this [1], as suggested by Eqn. 4, where $W^+ = A + B + 2^n - 1$.

$$S^+ = |A+B|_{2^n+1} = \begin{cases} \left| W^+ \right|_{2^{n+1}}, & \text{if } W^+ \ge 2^{n+1} \\ \left| W^+ + 2^n + 1 \right|_{2^{n+1}}, & \text{if } W^+ < 2^{n+1} \end{cases} \quad (4)$$

Based on Eqn. 4, modulo-($2^n + 1$) adders have been built that utilize an n-bit RPP or TPP adder, resulting in a latency of 2 log n + 9 and 2 log n + 6 UGD, respectively [1]. The TPP version of the adder, depicted in Fig. 2, uses additional preprocessing half-adders, four kinds of (h, g, p) cells, two varieties of parallel prefix cells, and doubled-up cells in some nodes of the TPP tree (see [1] for a description of all these cells). We have shown [4] that the special logic dedicated to computing the most-significant bit of the sum (i.e. s_8 in Fig. 2) does not always generate the correct result bit. This flaw can be corrected, with no speed penalty, by appropriately modifying the design. However, the required modifications introduce added complexity (see Fig. 3) that must be accounted for to achieve fairness in comparisons of cost and cost-effectiveness.

Table 1. Symbols used in our circuit diagrams.

Symbol name	Block diagram	Gate network
Carry operator		
h, *g*, *p* generator		
Carry-in incorporator		
Excess-1 half-adder		
Buffer		
g inverter		
Inverting swapper		

Modulo-(2^n + 1) addition with diminished-1 residue representation leads to a simpler TPP adder [13]. However, a separate zero handler (Fig. 4) is needed to take care of zero operands and zero result, where $z_a = 0$ ($z_b = 0$, $z_s = 0$) indicates that A (B, S) is zero. Let $A' = A - 1$, $B' = B - 1$, and $S' = S - 1$ denote the diminished-1 representations of the modulo-(2^n + 1) operands and sum, respectively. Also, let $W'' = (w''_n w''_{n-1} \ldots w''_1 w''_0)_{\text{two}} = A' + B' = A + B - 2$ be the double-diminished sum of A and B, decomposed into a single bit w''_n and the n-bit number $|W''|_{2^n}$, as before. Then, S' can be computed as in Eqn. 5, where the case of $W'' = A' + B' = 2^n - 1$, which corresponds to $S = 0$, has not been included; the zero handler unit of Fig. 4 will take care of the latter case.

$$S' = S - 1 = |A + B|_{2^n+1} - 1$$

$$= \left\{ \begin{array}{ll} \left| W'' \right|_{2^n}, & \text{if } W'' \geq 2^n \\ \left| W'' \right|_{2^n} + 1, & \text{if } W'' < 2^n - 1 \end{array} \right\} = |W''|_{2^n} + \overline{w''}_n \qquad (5)$$

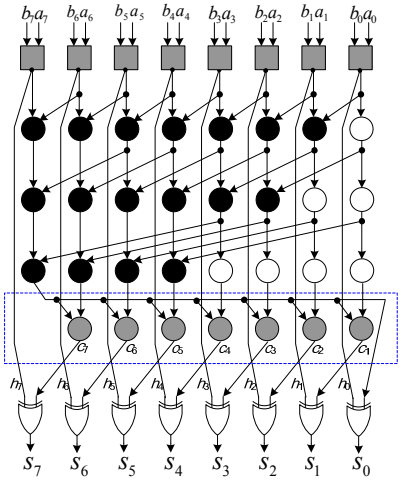


Fig. 1. Previously proposed designs for fast mod-(2^8 – 1) addition [2].

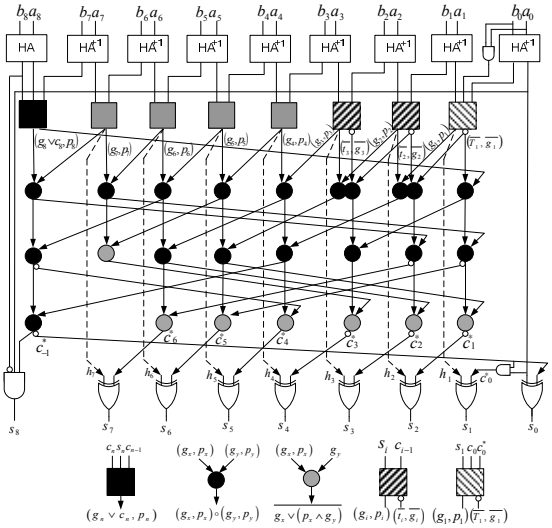


Fig. 2. Mod-257 TPP adder (Fig. 3 of [1], with outputs labeled s_i instead of r_i).

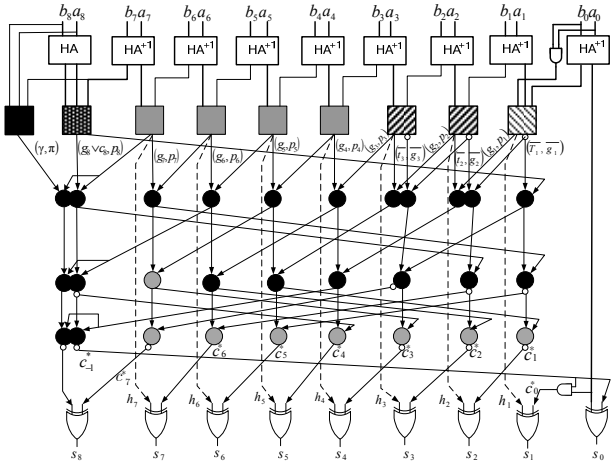


Fig. 3. The corrected mod-257 TPP adder.

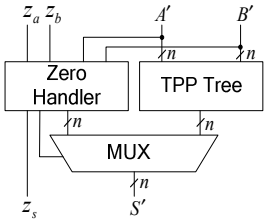


Fig. 4. Complete diminshed-1 mod-(2^n + 1) adder with zero handler.

The latency of the TPP tree in Fig. 4 is 2 log n + 3 UGD [13]. With 2 UGD for the final mux, the overall delay becomes 2 log n + 5 UGD. The prime advantage of the TPP or RPP implementation of mod-($2^n \pm 1$) adders is that it requires a single n-bit addition. Otherwise, post-addition increments would necessitate another n-bit full-carry-propagate addition.

Drawbacks of the TPP method include the use of a customized parallel-prefix adder that cannot be replaced by alternative n-bit adders. Thus, the exploration of the design space with regard to area/speed/power tradeoffs must be performed anew for each design.

In Section 4, we introduce generic mod-($2^n \pm 1$) adders with only one n-bit addition, leading to a number of benefits over the designs discussed in this section, including smaller area in some instances. We will show in Section 6 that the advantages gained from this generic implementation are well worth the speed tradeoff in one case.

3. Signed-LSB Representation

Modulo-($2^n + 1$) residues are in the range $[0, 2^n]$. There are two common representations for this range of integer values. The more "natural" of the two representations is the standard weighted binary one, whereby a residue R is represented as $(r_n r_{n-1} \ldots r_0)_{\text{two}}$, in which $r_n = 1$ only for $R = 2^n$. Therefore, this is not a faithful representation, because there are $2^n - 1$ codewords with $r_n = 1$ that do not represent valid residues. The second representation option, which is both faithful and leads to more efficient designs, is the diminished-1 representation.

With diminished-1 representation, a residue R is encoded as (z_r, R'), where $z_r = 0$ iff $R = 0$, and $R' = (r'_{n-1} r'_{n-2} \ldots r'_0)_{\text{two}} = R - 1$ when $R \geq 1$.

When a carry c enters position n (of weight 2^n) in mod-($2^n + 1$) addition, it can be reentered as $-c$ in position 0. Equation 6 justifies this *end-around borrow*.

$$|2^n c|_{2^n+1} = |(2^n + 1)c - c|_{2^n+1} = -c \qquad (6)$$

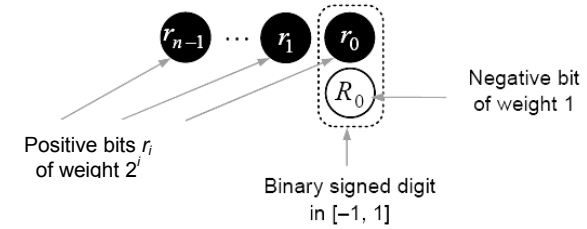


Fig. 5. Signed-LSB representation of mod-($2^n + 1$) residues.

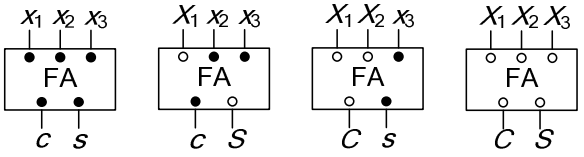


Fig. 6. The four functionalities of the same full-adder (FA) cell, with posibit and negabit inputs/outputs.

One way to avoid the costly subtraction (i.e., a second full-blown addition) is to simply store the reentrant borrow in position 0. Figure 5 depicts the signed-LSB representation of modulo-($2^n + 1$) residues in $[-1, 2^n - 1]$. The black circles hold lowercase variables and indicate normal bits (posibits). The single white circle, with an uppercase variable, represents a negated bit (negabit) [3]. The posibit and negabit in position 0 of Fig. 5 together form a binary signed digit in [–1, 1]; hence the designation "signed-LSB." The representation is faithful, because any possible bit assignment represents an integer value in $[-1, 2^n - 1]$, although most values in this range (including all even values) have two representations.

The coexistence of posibits and negabits in the same weighted position (or "column"), in general, calls for specialized adder cells, of the types used in the well-known Pezaris array multiplier [10]. However, if we use an inverted encoding for negabits, that is, let logical 0 (1) stand for the arithmetic value –1 (0), standard full-adder cells can handle any mix of equally weighted bits of either polarity exactly as if we had nothing but posibits [3]. The arithmetic value of a negabit X with inverted encoding equals $X - 1$. Figure 6 shows four possible combinations of positive and negative inputs for a full adder, where black (white) circles inside the FA block denote posibits (negabits). Note that the polarities of the sum and carry are determined by the minority and majority of input polarities, respectively. As an example, Eqn. 7 justifies the functionality of the second full adder from the right in Fig. 6, where $X_1 + X_2 + x_3 = 2c + s$ defines the standard functionality of a full-adder cell and $\|E\|$ denotes the arithmetic value of the algebraic expression E.

$$\|X_1 + X_2 + x_3\| = X_1 - 1 + X_2 - 1 + x_3 = 2c + s - 2 = \|2C + s\| \qquad (7)$$

4. New Modulo-($2^n \pm 1$) Adders

4.1. Modulo-($2^n + 1$) Adder

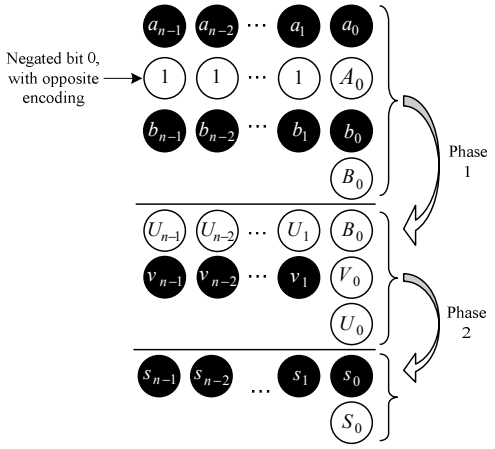
Given the signed-LSB representation of mod-($2^n + 1$) residues, as depicted in Fig. 5, mod-($2^n + 1$) addition can be performed according to Fig. 7a, where a 1-filled white circle denotes a negabit with arithmetic value 0, $U_i = \overline{a_i \oplus b_i}$, $v_{i+1} = a_i \vee b_i$ (for $1 \leq i \leq n - 1$), $U_0 = a_0 \oplus b_0 \oplus A_0$, $v_1 = a_0 b_0 \vee A_0 (a_0 \vee b_0)$, and $V_0 = \overline{v_n}$.

The transformation in phase 1 at the top of Fig. 7a can be implemented by an n-bit carry-save adder, where the leftmost carry v_n is stored, based on Eqn. 6, as the negabit V_0. Two methods can be envisaged for the transformation in phase 2 at the bottom of Fig. 7a.

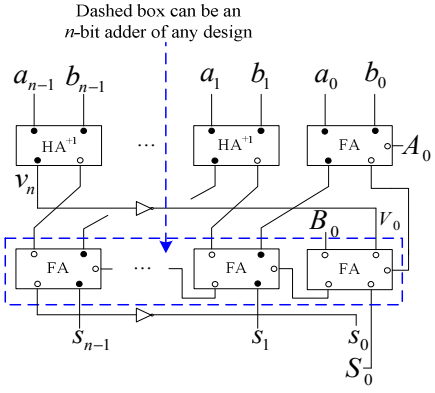
(a) Generic method: The phase-2 addition in Fig. 7a can be delegated to a standard n-bit adder, with the least-significant bit of the sum forming the negabit S_0

and the carry out S_n stored in position 0 as the normal bit s_0, again based on Eqn. 6. The dashed box in Fig. 7b encloses this generic adder, that can be replaced by any other adder meeting the design goals in terms of latency, area, and power. For example, Fig. 8a depicts an RPP realization of Fig. 7b using a Kogge-Stone [6] parallel prefix adder. The overall latency is 2 log n + 7 UGD, which is broken down thus: 1 UGD for the (p, g) cells, 2 for HA^{+1} cells, 2 for carry-in incorporation near the bottom, 2 for the final XOR gates, and 2 log n for the parallel prefix tree.

(b) TPP method: To avoid the delay of 2 UGD due to the carry-in incorporation below the parallel-prefix network of Fig. 8a, that is, to reduce the total latency to 2 log n + 5 UGD, we leave U_0 intact, as the negabit component of the final signed-LSB number, and feed the rest of the bits (i.e., the two n-bit rows) into a TPP tree as in Fig. 8b. This TPP structure is identical to the diminished-1 adder of [13].



(a) Computation in dot notation



(b) Generic circuit realization

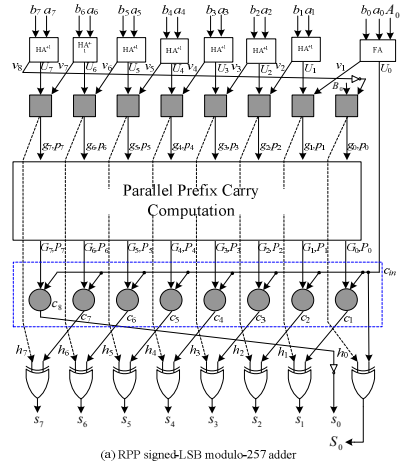
Fig. 7. Signed-LSB mod-(2^n + 1) addition process and its circuit realization.

4.2. Modulo-(2^n – 1) Adder

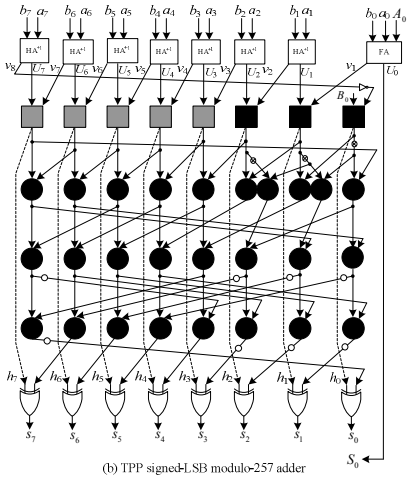
A carry c into position n of an n-bit mod-(2^n – 1) adder can be wrapped around and stored in position 0. This is justified by Eqn. 8.

$$|2^n c|_{2^n-1} = |(2^n - 1)c + c|_{2^n-1} = c \qquad (8)$$

If we use signed-LSB mod-(2^n – 1) residues, the same addition scheme as in Fig. 7a works, except that the stored bit in the middle segment will be a posibit v_0 as in Fig. 9a. This leads to a posibit as the least-significant output (s_0) of the generic adder. However, the carry-out of this adder will be stored as an inversely encoded negabit $S_0 = S_n$. Thus, the desired generic modulo-(2^n – 1) adder can be obtained by simply removing the two inverters from Fig. 7b (Fig. 9b). Likewise, the RPP mod-(2^n – 1) adder is obtained by removing the two inverters of Fig. 9a and inverting the polarity of the two least significant bits (i.e., regarding s_0 as S_0, and vice versa). For TPP implementation, we can use the less complex TPP tree of [5] (Fig. 1b).



(a) RPP signed-LSB modulo-257 adder



(b) TPP signed-LSB modulo-257 adder

Fig. 8. Mod-(2^n + 1) signed-LSB addition.

5. Conversion from/to Binary

Conventional binary-to-residue conversion methods for the moduli $2^n \pm 1$ (e.g., [15]) compute $X = |I|_{2^n-1}$ and $Z = |I|_{2^n+1}$, where I is an integer input and $X = (x_{n-1} \ldots x_1 x_0)_{\text{two}}$ and $Z = (z_n z_{n-1} \ldots z_1 z_0)_{\text{two}}$ are n-bit and $(n+1)$-bit residues, respectively. To find the signed-LSB representation of X, we attach the zero-valued negabit $X_0 = 1$ to form the signed LSB alongside x_0. For signed-LSB representation of Z, we let $Z_0 = \bar{z}_n$, based on the observation in Eqn. 6 and the assignment of V_0 in Fig. 7a. In both cases, conversion of input to residue representation is very simple.

For the reverse conversion, we show that the negabit components do not introduce any inefficiency. Fast residue-to-binary converters normally implement the Chinese remainder theorem [7] via carry-save adders (CSA) for the required multioperand addition. For example, with the 3-modulus set $\{2^n \pm 1, 2^n\}$, Eqn. 9, adapted from [11], converts $(X, Y, Z)_{\text{RNS}}$, where $X = |I|_{2^n-1}$, $Y = |I|_{2^n}$, and $Z = |I|_{2^n+1}$, to I as follows:

$$I = 2^n H + Y \qquad (9)$$
$$= 2^n [\,|-(2^n Y + Z) + 2^{n-1}(2^n+1)(X+Z)|_{2^{2n}-1}] + Y$$

Since $Y < 2^n$, only the computation of H is needed for finding I. To allow comparison, we present the formula that yields H for both standard and signed-LSB representations of X and Z, where a few intermediate steps of the derivation have been omitted for brevity ($\bar{Y}$ stands for $\bar{y}_{n-1} \ldots \bar{y}_1 \bar{y}_0$):

$$H = |-2^n(2^n-1-Y) - Z + 2^{2n-1}Z + 2^{n-1}Z + 2^{2n-1}X + 2^{n-1}X|_{2^{2n}-1}$$
$$= |2^n \bar{Y} + 2^{2n-1}X + 2^{n-1}X + 2^{n-1}Z + 2^n - 1 - 2^{2n-1}Z|_{2^{2n}-1} \qquad (10)$$

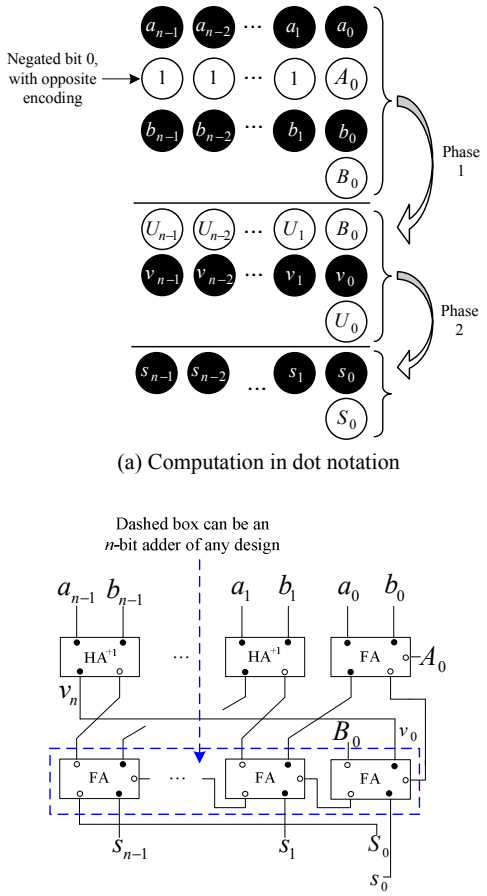
Denoting the negative term $|-2^{2n-1}Z|_{2^{2n}-1}$ as N_w and N_s for weighted $(n+1)$-bit and signed-LSB representations of Z (according to Eqns. 11 and 12), respectively, we obtain the following. Notationally, $\bar{Z}_w$ and $\bar{Z}_s$ stand for $\bar{z}_n \bar{z}_{n-1} \ldots \bar{z}_1 \bar{z}_0$ and $\bar{z}_{n-1} \ldots \bar{z}_1 \bar{z}_0$, respectively.

$$N_w = |-2^{2n-1}(2^{n+1} - 1 - \bar{Z}_w)|_{2^{2n}-1} \qquad (11)$$
$$= |-2^{3n} + 2^{2n-1} + 2^{2n-1}\bar{Z}_w|_{2^{2n}-1}$$
$$N_s = |-2^{2n-1}(2^n - 1 - \bar{Z}_s + Z_0 - 1)|_{2^{2n}-1} \qquad (12)$$
$$= |-2^{n-1} + 2^{2n-1} + 2^{2n-1}(\bar{Z}_s + \bar{Z}_0)|_{2^{2n}-1}$$

Plugging Eqns. 11 and 12 into Eqn. 10 produces the following expressions:

$$H_w = |2^n \bar{Y} + 2^{2n-1}X + 2^{n-1}X + 2^{n-1}Z + 2^{2n-1}\bar{Z}_w + 2^{2n-1} - 1|_{2^{2n}-1} \qquad (13)$$

$$H_s = |2^n \bar{Y} + 2^{2n-1}X + 2^{n-1}X + 2^{n-1}Z + 2^{2n-1}\bar{Z}_s + 2^{2n-1}\bar{Z}_0 + 2^{2n-1} - 2^{n-1}|_{2^{2n}-1} \qquad (14)$$



(a) Computation in dot notation

(b) Generic circuit realization

Fig. 9. Signed-LSB mod-(2^n – 1) addition process and its circuit realization.

2^{2n-1}	2^{2n-2}	…	2^n	2^{n-1}	2^{n-2}	…	2^0
$\bar{y}_{n-1}$	$\bar{y}_{n-2}$	…	$\bar{y}_0$	x_0	x_{n-1}	…	x_1
x_0	x_{n-1}	…	x_1	$\bar{z}_n$	$\bar{z}_{n-1}$	…	$\bar{z}_1$
z_n	z_{n-1}	…	z_1	z_0	1	…	1
$\bar{z}_0$	1	…	1	1			

(a) Weighted representation

2^{2n-1}	2^{2n-2}	…	2^n	2^{n-1}	2^{n-2}	…	2^0
$\bar{y}_{n-1}$	$\bar{y}_{n-2}$	…	$\bar{y}_0$	x_0	x_{n-1}	…	x_1
x_0	x_{n-1}	…	x_1	z_0	$\bar{z}_{n-1}$	…	$\bar{z}_1$
$\bar{z}_0$	z_{n-1}	…	z_1	X_0			
X_0	1	…	1	Z_0			
$\bar{Z}_0$				1			

(b) Signed-LSB representation

Fig. 10. Computing H_w and H_s via multioperand addition, as specified by Eqns. 13-14.

Figure 10 shows how H_w (H_s) is computed by means of a 4-operand (5-operand) mod-($2^{2n} - 1$) addition. Note that computing the final sum for both parts of Fig. 10 requires two CSA levels, before the final $2n$-bit mod-($2^{2n} - 1$) addition. The reason is that the two 5-deep columns in part b are preceded by shallower columns; otherwise three CSA levels would have been required.

6. Comparisons and Applications

Table 2 shows the latency of different mod-($2^n \pm 1$) adders in terms of UGD. Note that the diminished-1 entry includes 2 UGD for the multiplexer in Fig. 4.

From Table 2, we see that with gate-level analysis, modulo-($2^n \pm 1$) adders based on the signed-LSB representation are quite competitive with the best available designs. The fact that the mod-($2^n - 1$) adder is slightly slower is of no consequence in RNS applications, given that the speed of arithmetic is dictated by the slower mod-($2^n + 1$) channel. Even at this coarse level, the signed-LSB approach offers several advantages. The use of a generic, rather than a custom-designed, fast adder reduces the design time, simplifies debugging and modifications, and facilitates area/time/power tradeoffs. Furthermore, removal of the zero-handler circuitry, needed for the diminished-1 representation, and simpler conversions from/to binary, give an edge to the signed-LSB approach. And all this is before we take the area and power advantages of a more regular design into account.

To assess this latter advantage, we produced VHDL code for all the adders listed in Table 2 and ran simulations and synthesis, for n = 4, 8, and 16, using the Synopsys Design Compiler. The target library was based on TSMC 0.13 μm standard CMOS technology. The results appear in Table 3. Note that the corrected version of the adder in [1], depicted in Fig. 3, is used and that delay and area figures cited for diminished-1 adders include the zero-handler logic.

Besides the advantages discussed above, the unified design strategy for modulo-($2^n \pm 1$) adders allows us to synthesize reconfigurable adders that can process inputs for different moduli, based on configuration signals provided by control circuitry. The design of such an adder entails replacing some of the inverters in our original designs with multiplexers. As is usually the case for reconfigurable logic, a latency penalty is paid for this capability.

Such a reconfigurable modular adder (RMA) allows fault-tolerant designs with much lower hardware redundancy than full replication. With three moduli, say, we might realize four such adders and then configure them to perform the three different modular additions ($2^n - 1$, 2^n, and $2^n + 1$), with one of them kept as a spare (see Fig. 11). Upon a detected fault in one of the available channels, the spare can be brought in and the good adders configured accordingly.

In fact, the scheme of Fig. 11 can be improved as follows, given that reconfiguration occurs via shift switching. Full recovery in the presence of one faulty RMA would still be possible if each RMA were 2-way, rather than 3-way, reconfigurable: the two RMAs on the left of Fig. 11 are capable of switching between mod-($2^n - 1$) and mod-2^n operation, while the two RMAs on the right can switch between mod-(2^n) and mod-($2^n + 1$) only. Such two-way reconfigurable RMAs would be both faster and simpler than the corresponding three-way designs.

Table 2. Latency of mod-($2^n \pm 1$) adders in UGD.

Representation	Modulus	RPP	TPP	Ref.
Weighted	$2^n + 1$	2 log n + 9	2 log n + 6	[1]
Diminished-1	$2^n + 1$	2 log n + 7	2 log n + 5	[9]
Signed-LSB	$2^n + 1$	2 log n + 7	2 log n + 5	New
Weighted	$2^n - 1$	2 log n + 5	2 log n + 3	[5]
Signed-LSB	$2^n - 1$	2 log n + 7	2 log n + 5	New

Table 3. TSMC 0.13 μm synthesis results for mod-($2^n \pm 1$) adders, with n = 4, 8, or 16.

Representation	Modulus	Delay (ns)	Critical path	Area (μm^2)
Weighted	$2^4 + 1$	0.4214	$a_4 \rightarrow s_1$	1618
Diminished-1	$2^4 + 1$	0.2935	$a_0 \rightarrow s_0$	1346
Signed-LSB	$2^4 + 1$	0.3003	$a_0 \rightarrow s_3$	1402
Weighted	$2^4 - 1$	0.2820	$a_2 \rightarrow s_2$	1173
Signed-LSB	$2^4 - 1$	0.3075	$a_0 \rightarrow s_0$	1292
Weighted;	$2^8 + 1$	0.5117	$b_5 \rightarrow s_6$	4793
Diminished-1	$2^8 + 1$	0.4998	$b_3 \rightarrow s_2$	3498
Signed-LSB	$2^8 + 1$	0.4999	$b_1 \rightarrow s_1$	3678
Weighted	$2^8 - 1$	0.3999	$a_0 \rightarrow s_5$	2299
Signed-LSB	$2^8 - 1$	0.4998	$a_0 \rightarrow s_6$	3161
Weighted	$2^{16} + 1$	0.6207	$a_{16} \rightarrow s_7$	6927
Diminished-1	$2^{16} + 1$	0.5791	$a_{13} \rightarrow s_{12}$	6377
Signed-LSB	$2^{16} + 1$	0.5803	$b_4 \rightarrow s_5$	6733
Weighted	$2^{16} - 1$	0.4868	$a_{13} \rightarrow s_5$	5850
Signed-LSB	$2^{16} - 1$	0.5901	$a_9 \rightarrow s_8$	6243

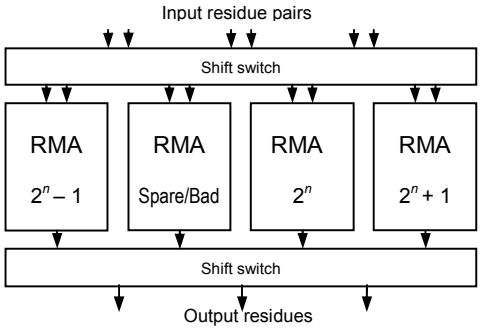


Fig. 11. Fault-tolerant RNS processor using reconfigurable mod-(2^n–1)/2^n/(2^n+1) adders.

Note that the spare channel in Fig. 11 need not sit idle during normal operation before the occurrence of the first fault. It can be utilized to check the correct functioning of one of the other channels through comparison, with the channel checked chosen on a cyclic basis (additional switching mechanisms and a comparator must be added to the hardware).

It is also possible to use a fault tolerance scheme based on time redundancy, in lieu of, or on top of, the scheme shown in Fig. 11. Consider the possibility of two faulty channels among the four in Fig. 11, or an implementation that does not have any redundant channel. Then, the existing or surviving channels can be used on a time-multiplexed basis to perform the tasks required of all three channels. The computation now takes longer to complete, but the system is allowed to degrade gracefully, rather than fail abruptly.

7. Conclusion

We have shown that representing residues in redundant form, with a binary signed-digit in the LSB position and ordinary bits in all other positions, allows us to perform mod-$(2^n - 1)$ and mod-$(2^n + 1)$ addition by means of a generic binary (mod-2^n) adder that is preceded by a carry-save adder and augmented with a very small amount of additional logic. This approach enables the use of predesigned building blocks, thus leading to easier/faster exploration of the design space, simpler testing/verification, and greater confidence in the correctness of the resulting design.

The similarity of our modular adder designs for the moduli $2^n - 1$ and $2^n + 1$, and their incorporation of a standard mod-2^n adder, permits the design of a flexible modular adder that can perform mod-$(2^n - 1)$, mod-2^n, or mod-$(2^n + 1)$ addition with appropriate control settings. Such a reconfigurable adder is useful in time-multiplexed use of RNS arithmetic mechanisms (particularly for input and output conversions that are not extensively utilized) and allows the design of fault-tolerant arithmetic units with fairly low redundancy.

We plan to pursue two avenues in continuing this research. The first is to tackle subtraction. As a rule, subtraction can be converted to addition by means of negation (sign change). However, this leads to some difficulty when standard weighted representation is used with mod-$(2^n + 1)$ subtraction. This is because:

$$D^+ = |A-B|_{2^n+1} = |A + 3 + (2^{n+1} - 1) - B|_{2^n+1} = |A+B+3|_{2^n+1}$$

Use of the same addition scheme as in Eqn. 4 would produce an $(n + 2)$-bit result for W^+, thereby leading to considerable overhead. Our signed-LSB method does not suffer from this problem. The second avenue is developing designs for mod-$(2^n \pm 1)$ multipliers. We believe that signed-LSB representation might have an edge over diminished-1 representation [2] for multiplication. This results, in essence, from avoiding the different treatment of zero and nonzero multiples of the multiplicand. As in the case of adders, configurable modular multipliers are possible and beneficial.

Acknowledgments

We thank Saeed Nejati for his assistance in obtaining the synthesis results, and Hanieh Alavi for her catching of the design flaw in [1]. G. Jaberipur's research was supported in part by the IPM School of Computer Science (under Grant # CS 1387-3-01), and in part by Shahid Beheshti University (under Grant # D/600/212).

References

[1] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast Parallel-Prefix Modulo 2^n+1 Adder", *IEEE Trans. Computers*, Vol. 53, No. 9, pp. 1211-1216, 2004.

[2] C. Efstathiou, *et al.*, "Efficient Diminished-1 Modulo 2^n + 1 Multipliers," *IEEE Trans. Computers*, Vol. 54, No. 4, pp. 491-496, 2005.

[3] G. Jaberipur, B. Parhami, and M. Ghodsi, "Weighted Two-Valued Digit-Set Encodings: Unifying Efficient Hardware Representation Schemes for Redundant Number Systems," *IEEE Trans. Circuits and Systems I*, Vol. 52, No. 7, pp. 1348-1357, 2005.

[4] G. Jaberipur and H. Alavi, "Comment on 'Fast Parallel Prefix Modulo 2^n+1 Adder'," on-line supplement to this paper, available at: http://www.ece.ucsb.edu/~parhami/pubs_folder/parh09-arith19-supplement.pdf

[5] L. Kalamboukas, *et al.*, "High-Speed Parallel-Prefix Modulo $2^n - 1$ Adders," *IEEE Trans. Computers*, Vol. 49, No. 7, pp. 673-680, 2000.

[6] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Trans. Computers,* Vol. 22, No. 8, pp. 786-792, 1973.

[7] S.-H. Lin and M.-H. Sheu, "VLSI Design of Diminished-One Modulo 2^n + 1 Adder Using Circular Carry Selection," *IEEE Trans. Circuits and Systems II*, Vol. 55, No. 9, pp. 897-901, 2008

[8] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*, Imperial College Press, 2007.

[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford Univ. Press, 2nd ed., 2009.

[10] Pezaris S.D., "A 40 ns 17-bit Array Multiplier," *IEEE Trans. Computers*, Vol. 20, No. 4, pp. 442-447, 1971.

[11] M. A. Soderstrand, *et al.*, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.

[12] A. Tyagi, "A Reduced-Area Scheme for Carry-Select Adders," *IEEE Trans. Computers*, Vol. 42, No. 10, pp. 1163-1170, 1993.

[13] H. T. Vergos *et al.*, "Diminished-One Modulo 2^n + 1 Adder Design," *IEEE Trans. Computers,* Vol. 51, No. 12, pp. 1389-1399, 2002.

[14] B. Vinnakota and V. V. Bapeswara Rao, "Fast Conversion Techniques for Binary-Residue Number Systems," *IEEE Trans. Circuits and Systems I*, Vol. 41, No. 12, pp. 927-929, 1994.

[15] Z. Wang, G. A. Jullien, and W. C. Miller, "An Improved Residue-to-Binary Converter," *IEEE Trans. Circuits and Systems I,* Vol. 47, No. 9, pp. 1437-1440, 2000.