ON MODULO (2^n+1) ARITHMETIC LOGIC

Dharma P. Agrawal
Department of Electrical and Computer Engineering
Wayne State University
Detroit, Michigan 48202

and

T.R.N. Rao
Computer Science Department
Southern Methodist University
Dallas, Texas 75275

Abstract

A novel format for representing modulo (2^n+1) numbers, is shown to be helpful in achieving modular addition and complementation. Logic for fast addition using carry-look-ahead and modular complementation is also presented.

Introduction

Application of modular arithmetic in error diagnosis and residue computers is well established.[1-5] The residue number system modulo 2^n and (2^n-1) are beneficial if they are prime pairwise. The addition modulo 2^n or (2^n-1) can be accomplished by n-bit binary adder. The carry overflow is thrown away in the first case while end-around-carry is used for mod 2^n-1. Recently, increased interest has been shown in moduli of the form (2^n+1).[6,7] The logic implementation of modulo (2^n+1) addition is not that simple and in the past the necessity of more complex design approach has discouraged its use. The inclusion of mod (2^n+1) provides more flexibility to the system design. This paper introduces a novel way of representing mod (2^n+1) numbers and of reducing the design complexity considerably, thereby making (2^n+1) as a serious candidate for modular operations.

Number Representation

Let Z_m denote the set of integers {0,1,2, ... m-1} called here the number system of modulus m. For $m = 2^n+1$, numbers in Z_m cannot be represented as binary n-tuples. Therefore $X \varepsilon Z_m$ is represented by a binary (n+1)-tuple $\underline{X}$ of the form

$$\underline{X} = (x_{n-1} \cdots x_i \cdots x_o,\ I_x) \qquad (1.a)$$

Where x_i has the usual weight of 2^i and I_x has a weight of 1, the same as x_o, I_x is called the zero indicator and equals zero iff X = 0.

Hence
$$X = I_x \left[\sum_{i=o}^{n-1} x_i 2^i + 1 \right] \qquad (1.b)$$

Setting

$$x = I_x \sum_{i=0}^{n-1} x_i 2^i, \quad \text{for } o \le x \le m-2 = 2^n-1 \qquad (1.c)$$

We have $X = I_x .(x+1)$ (2)

This representation is utilized throughout this paper.

Modular Addition

Let S be the addition modulo m of two numbers X and $Y \varepsilon Z_m$. To find s, let us define Q and C_n as

$$Q = 1 \text{ iff } x+y \ge m-1 \qquad (3.a)$$

and

$$C_n = 1 \text{ iff } x+y+I_x I_y \ge m-1 \qquad (3.b)$$

It is worth mentioning that Q can be obtained as an overflow when x and y are added by n-bit parallel adder. Similarity, C_n is detectable by the overflow of n-bit adder with "hot input" $I_x I_y$ adder to x and y as a carry-in to the lowest-significant bit (l.s.b.) position.

At this point, s and I_s for different numerical range of X and Y can be computed as is illustrated in Table 1. The arithmetic expressions for ($I_x + I_y$) and S can be easily obtained as

$$I_x + I_y = I_x I_y + I_x \vee I_y \qquad (4.a)$$

and
$$\begin{aligned} S = I_s(s+1) &= |X+Y|_m \\ &= |x+y+I_x + I_y|_m \\ &= (I_x \vee I_y)|x+y+I_x I_y+1|_m \qquad (4.b) \end{aligned}$$

It may be noted that v represents a logical OR operation while + denotes an arithmetic addition and $|a|_m$ represents the smallest non-negative integer congruent to a modulo m.

The expression (4.b) can be utilized to formulate s and I_s for the different cases shown in Table 1 and can be given as:

Cases i, ii
$$s = |x+y+I_xI_y|_{m-1} = x+y+I_xI_y \quad \text{and} \quad I_s = I_x \vee I_y \tag{5.a}$$

Cases iii
$$s = |x+y+I_xI_y|_{m-1}=0 \quad \text{and} \quad I_s = \overline{C}_n=0 \tag{5.b}$$

Case iv
$$s = |x+y|_{m-1} = x+y+1-m \quad \text{and} \quad I_s = I_x \vee I_y \tag{5.c}$$

It can be readily seen that relations (5) can be represented by a flow diagram shown in Fig. 1. In this diagram $\lfloor a \rfloor$ represents the integer part of a. Figure 2 illustrates a straightforward scheme utilizing an n-bit fast carry-look-ahead binary adder. Initially, the carry-in to l.s.b. (least significant bit) is kept zero by setting the D-Flip Flop. Once the carry-overflow is available, its value is stored in D-Flip Flop and a second ADD cycle time is allowed. We obtain a logical expression for I_s as follows:

$$I_x = (Q \vee \overline{C}_n)(I_x \vee I_y) \tag{6}$$

where Q and C_n have been defined by relations 4.a and 4.b. They are also indicated in Figs. 1 and 2.

The technique suggested in the reference (6) needs larger number of bits to allow the needed redundancy. Different alternatives proposed by Chinal[7] requires binary addition (or G and P-term generation) followed by one or more stages of output correction networks. This way, the scheme given here in Fig. 2 is much simpler than those available in the literature[6,7]. Moreover it can be implemented by off-the-shelf integrated circuits.

Carry-Look-Ahead (CLA) Adder

As already pointed out, the scheme of Fig. 2 requires two cycles of addition operations plus a D-Flip Flop and a properly delayed clock pulse. One way to perform addition in one step is to compute two conditional sums; the first one as the addition of x and y and second as the sum of x, y and the "hot bit" I_xI_y as carry-in at the l.s.b. position. Then the signals Q and C_n can be utilized to dictate a proper choice between them. Such an arrangement requires 2-sets of n-bit adders and n-multiplexers. Another alternative is to design carry lookahead circuits for direct modular arithmetic. The design procedure for such a direct addition scheme is given below.

To achieve look-ahead, first of all, carry generate and propagate terms have to be computed. For a binary adder, let G_i and P_i respectively denote the carry generate and propagate terms at the i^{th} bit position when two numbers a and b are added.

Then
$$G_i \equiv a_{i-1}\, b_{i-1} \tag{7.a}$$

$$P_i \equiv a_{i-1} \oplus b_{i-1} \tag{7.b}$$

so that
$$\begin{aligned} C_i &= G_i \vee P_iC_{i-1} \\ &= G_i \vee P_iG_{i-1} \vee P_iP_{i-1}G_{i-2} \\ &\qquad \vee P_iP_{i-1} \cdots P_2P_1C_0 \end{aligned} \tag{7.c}$$

and
$$S_i = P_{i+1} \oplus C_i \tag{7.d}$$

where v and $\oplus$ are Logical-OR and Exclusive-OR operations respectively and C_o is the carry-in at the l.s.b. position.

As clear from the discussions in the previous sections, modular C.L.A. logical relations can be obtained by first computing C_n with $C_0 = 0$ and then evaluating $C_0 = I_xI_y\overline{C}_n$. This substitution and a little manipulation gives C_i as $C_i = G_i \vee P_iP_{i-1} \vee P_iP_{i-1}G_{i-2} \vee \cdots\cdots \vee P_iP_{i-1} \cdots P_2G_1$

$$\begin{aligned} &\vee P_iP_{i-1} \cdots P_2P_1I_xI_y[\overline{G}_n\overline{P}_n \vee \overline{G}_n\overline{G}_{n-1}\,\overline{P}_{n-1} \vee \cdots \\ &\cdots \vee \overline{G}_n\overline{G}_{n-1}\overline{G}_{n-2} \cdots \overline{G}_{i+3}\overline{G}_{i+2}\overline{P}_{i+2} \vee \overline{G}_n\overline{G}_{n-1} \cdots \\ &\cdots \overline{G}_{i+2}\overline{G}_{i+1}] \end{aligned} \tag{8.a}$$

$$\text{and} \quad I_s = \overline{G}_n\overline{G}_{n-1}\overline{G}_{n-2} \quad \overline{G}_2\overline{G}_1P_nP_{n-1} \quad P_2P_1I_xI_y \tag{8.b}$$

Logical implementation of these relations give the modular carry-look-ahead unit.

Modular Complementation

Let Y be the modulo m complement of X. Then by definition

$$Y = I_x|m-x-1|_m = I_x|2^n-x|_m = I_y(y+1) \tag{9.a}$$

Hence
$$y = I_x|2^n-1-x|_m \tag{9.b}$$

and
$$I_y = I_x \tag{9.c}$$

When $I_x = 1$, the relation (9.b) is nothing but 1's complement of x and can be obtained simply by complementing individual bits of x. This leads to an obvious simple circuit implementation of relations (9.b) and (9.c) and is given if Fig. 3.

Concluding Remarks

The specific way of representing numbers presented here makes modulo (2^n+1) addition and complementation much simpler than available in the literature[6,7]. An added advantage is that available binary adders can be effectively utilized.

Acknowledgements

This work is supported by NSF grant ENG 76-11237 and Office of Naval Research Contract N00014-77-C-0455.

References

1. GARNER, H. L.: 'The Residue Number System,' IRE Trans., 1959, EC-8, 140-147.

2. SZABO, N. S., and TANAKA, R. I.: 'Residue Arithmetic and its Applications to Computer Technology,' McGraw-Hill, New York, 1967.

3. MASSEY, J. L., and GARCIA, O. N.,: Error Correcting Codes in Computer Arithmetic, Chapter 5 'Advances in Information Systems Science,' (Ed. J. L. Tou) Vol. 4, Plenum Press, New York, 1971, pp. 273-326.

4. RAO, T.R.N.: 'Error Coding for Arithmetic Processors,' Academic Press, New York, 1974.

5. SELLERS, F. F. Jr., HSIAO, M. Y., and BEARNSON, L. W.: 'Error Detecting Logic for Digital Computers,' McGraw-Hill, New York, 1968.

6. RAO, T.R.N., and TREHAN, A. K.: 'A Redundant Code for Modulo 2^k+1 Arithmetic,' First Southeastern Symposium on System Theory, Virginia Polytechnic Institute, May 1979, 11 pages.

7. CHINAL, J. P.: 'The Logic of Modulo 2^k+1 Adders,' 3rd Symposium on Computer Arithmetic, S.M.U., Dallas, Nov. 1975, Vol. IEEECS 75CH1017-3C, pp. 126-135.

Table 1

s and I_s for different range of X and Y

Case	Value of X+Y	I_x, I_y	x+y	Identification	I_s	S
(i)	0	0,0	0	$Q=0, C_n=0, I_x \vee I_y=0$	0	0
(ii)	[1,m-1]*	not both zero	[0,m-2]	$Q=0, C_n=0$	1	$x+y+I_xI_y$
(iii)	m	1,1	m-2	$Q=0, C_n=1$	0	0
(iv)	[m+1,2m-2]	1,1	[m-1,2m-4]	$Q=1, C_n=1$	1	(x+y+1)-m

* The square bracket [ ] indicates the range.

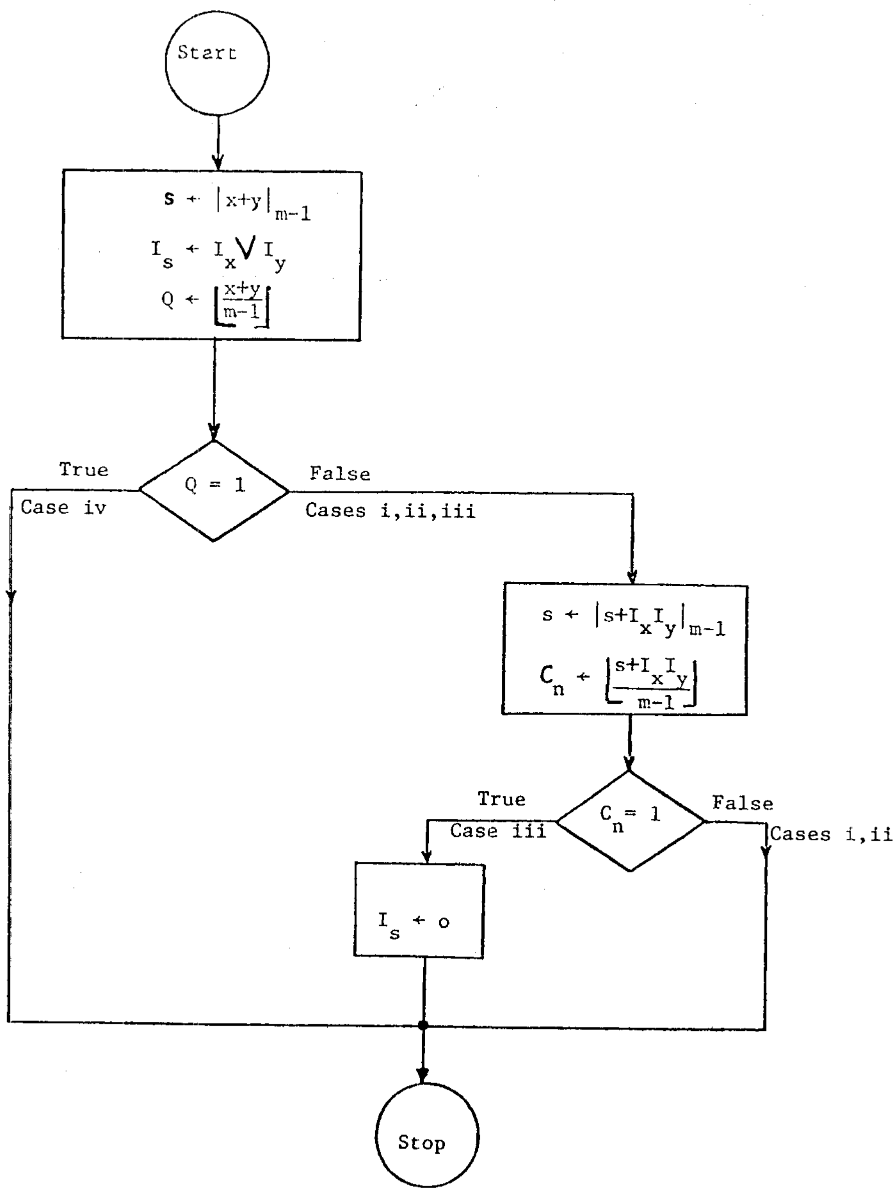

Fig. 1. Flow Diagram for Mod 2^n+1 Addition

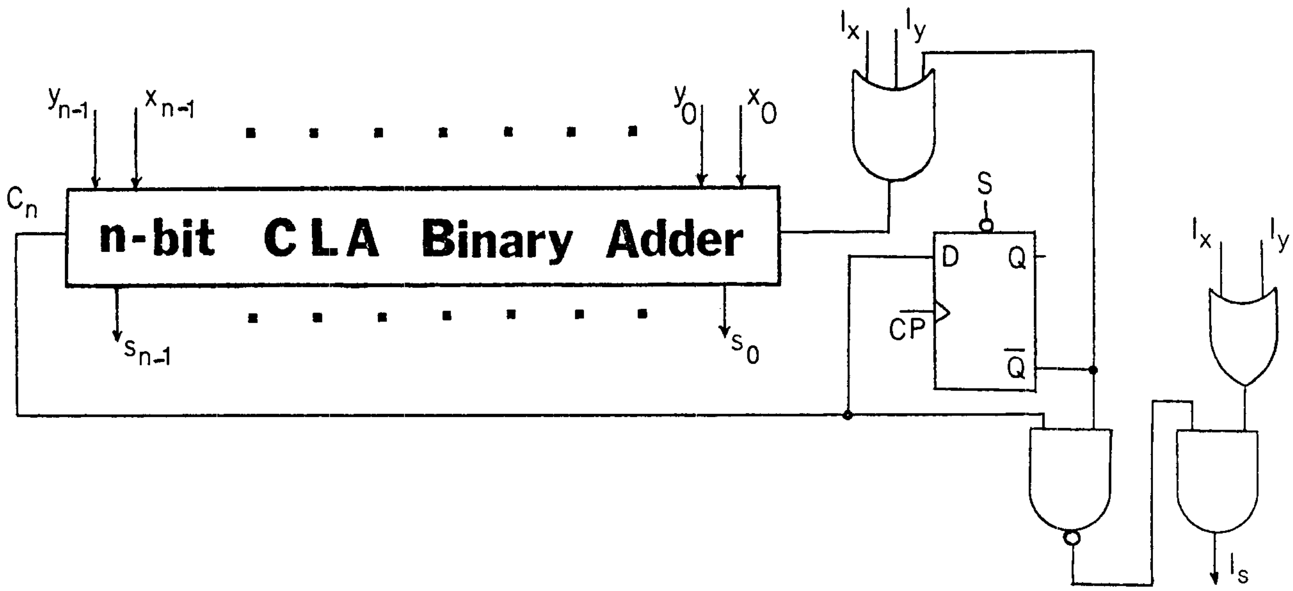

Fig. 1.b Logic diagram for Mod 2^n+1 addition.

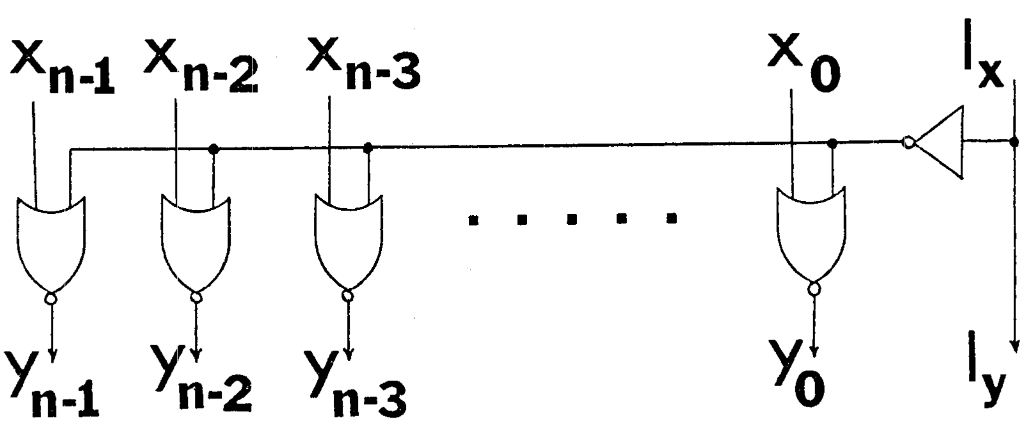

Fig 2. Modulo 2^n+1 complementation.