An Area-reduced Scheme for Modulo 2^n-1 Addition/Subtraction

Shaoqiang Bi*, Warren J.Gross*, Wei Wang***, Asim Al-Khalili**and M. N. S. Swamy**
*Department of Electrical & Computer Engineering, McGill University
**Department of Electrical & Computer Engineering, Concordia University
*** Department of Electrical & Computer Engineering, Purdue University Indianapolis
Email: shaoqiang.bi@mail.mcgill.ca, wjgross@ece.mcgill.ca, ww3@iupui.edu,
asim@ece.concordia.ca, swamy@ece.concordia.ca

***Abstract*-- In this paper, we present a versatile area-reduced scheme for modulo 2^n–1 adders and subtractors using a novel MUX-based increment/decrement algorithm. A FPGA-based comparison of the proposed modulo adder and the conventional modulo adder designs is carried out. The implementation results show that the proposed adder reduces the area close to 30% compared with the modulo adder of Bayoumi et al.. The delay and the power are also reduced around 10%. In addition, it is also shown that the proposed design requires less hardware resources than the parallel-prefix modulo adder of Kalampoukas et al. while providing a comparable operation speed.**

I. INTRODUCTION

Arithmetic modulo 2^n–1 (Mersenne numbers) has found many applications in the residue number systems (RNS) [1], cryptography [2] and fault-tolerant computer systems [3]. Efficient and fast modulo adders are a prerequisite for the corresponding high performance circuits in these fields. The standard implementation of a modulo 2^n–1 adder uses a conventional binary adder with the carry-out fed back to the carry-in to achieve the end-around carry. If ordinary adders are used, where the carry-out depends on the carry-in, a combinational loop is created that may lead to an unwanted race condition. Races 50 times longer than the normal addition time have been reported in [4]

A variety of researchers have investigated the problem of modulo 2^n–1 adder design [4]-[8]. In [5], a typical modular 2^n–1 adder design that utilizes two level binary adders and a multiplexer is described. When the modulo adder in [5] is implemented as carry look-ahead adders (CLA), it has a similar implementation speed with the CLA adder that carries out modulo 2^n–1 addition in two cycles. In [6], an approach for fast modulo 2^n–1 addition based on a modification of the traditional CLA is demonstrated. There, the logic formula for the carry-out is re-substituted as carry-in in the logic formulae for the sum bits. The savings in area and delay are reported for small operand lengths ($n \le 8$). It can be observed that the savings decrease fast when the operand length increases from n=2 to n=8 in [6, Table I]. Even faster designs based on the parallel-prefix carry computation approach have appeared recently in [7] – [8]. These modulo adders use special structures rather than conventional adders. Although improvement in terms of speed has been reported, we have to note though, that these parallel-prefix designs have complex interconnection and lead to efficiency problems if implemented directly in hardware, such as the large fan-out requirements of the Sklansky prefix structures and the complex wiring problems of the Kogge-Stone prefix structures. Alternative parallel-prefix structures have been proposed in the literature to solve the implementation problems [8], however they are also expensive.

In this paper, we present a versatile area-reduced scheme of modulo 2^n–1 adders and subtractors using a MUX-based increment/decrement algorithm. The design and FPGA implementation of the proposed modulo adder and the conventional modulo adder in [5] are carried out. The implementation results show that the proposed modulo adder reduces the area close to 30%. And the delay and the power are also reduced around 10%. In addition, it is also shown that the proposed design requires less hardware resources than the parallel-prefix modulo adders in [8] while providing a comparable operation speed.

II. PROPOSED INCREMENTER/DECREMENTER

In this section, we present a fast MUX-based incrementer/decrementer avoiding the delay introduced by the inherent ripple carry of the adder-based design.

A. Proposed MUX-based Decrement Algorithm

Let the input of a decrementer be $X = X_{n-1}\ldots X_1 X_0$. To find the output, start from the least significant bit (LSB) X_0 and search to the most significant bit (MSB) X_{n-1} for the first occurrence of bit '1'. Let the first bit '1' to be X_J. Then the output is obtained as follows:

1) *Complement all data bits X_I for I = J, J-1,…, 0.*
2) *Leave all other bits of X_I for I > J as they are.*
3) *Complement all bits if X = 0.*

Proposition 1 can also describe the proposed algorithm.

Proposition 1 Given any n-bit binary input X, we get its decrement result $Y = X$-1 as follows.

$$Y = X - 1 \Leftrightarrow \begin{cases} Y_j = \overline{X_j \oplus (X_0 + X_1 + \cdots + X_{j-1})}, & 1 \le j \le n-1 \\ Y_j = \overline{X_0}, & j = 0 \end{cases} \quad (1)$$

(Proof is omitted due to lack of space).

The following example is used to demonstrate the operation of the proposed algorithm.

Example 1: Find the decrement result of the 8-bit number $(10110100)_2$.

Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05)
1098-8068/05 $20.00 © 2005 **IEEE**

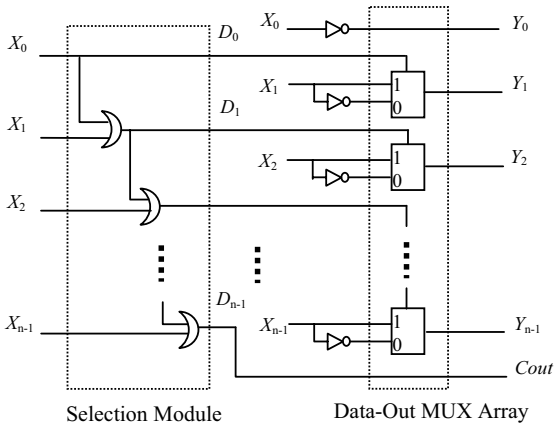

Fig. 1. The proposed MUX-based binary decrementer.

We carry out the operation by searching from the LSB to the MSB to find the first '1' bit which is the 3rd bit from the LSB. Thus, the first 3 LSBs 100 will be inverted to 011 in the output and the rest of the bits will keep their values as 10110 in the output. Then, the final output will be 10110011, which is exactly the desired result.

B. Proposed MUX-based decrementer

Based on the proposed algorithm, a new *n*-bit MUX-based decrementer is designed as shown in Fig. 1. It is composed of a data-out MUX array and a selection module (SM) used to find the first one bit. The output of SM is $D_0\ D_1 \cdots D_{n-1}$. When $D_J = 0$ (J=0,…,n-1), the input bits from X_J to X_0 are 0. From Proposition 1, it can be noted that each bit of the decrement result Y except Y_0 can be derived by a MUX operation. For example, by equation (1), we have $Y_1 = X_1X_0 + \overline{X_1}\,\overline{X_0} = \overline{X_1 \oplus X_0}$ that can be implemented by a MUX whose inputs are X_1 and $\overline{X_1}$ with a select signal connected to X_0.

C. Proposed MUX-based incrementer

The proposed MUX-based decrementer can be used to build a binary incrementer. The increment function will require that all the input bits be inverted in advance and then be sent to the proposed decrement circuit. Then, the inverted output of the decrementer will give the desired increment result. This feature can be summarized as Proposition 2.

Proposition 2 Given any *n*-bit binary number X, we have

$$X + 1 = \overline{\overline{X} - 1}$$

(Proof is omitted due to lack of space).

In Proposition 3, we show how to design a MUX-based incrementer using the proposed decrement algorithm.

Proposition 3 Given any *n*-bit binary input X, we get its binary increment result $Y = X+1$ as follows.

$$Y = X + 1 \Leftrightarrow \begin{cases} Y_j = \overline{X_j \oplus (\overline{X_0} + \overline{X_1} + \cdots + \overline{X_{j-1}})}, & 1 \le j \le n-1 \\ Y_j = \overline{X_0}, & j = 0 \end{cases} \quad (2)$$

(Proof is omitted due to lack of space).

Compared to the decrementer in Fig. 1, the only difference is that the incrementer has the input of its SM inverted.

III. PROPOSED MODULO ADDER/SUBTRACTOR

In modulo 2^n-1 adders and subtractors, the carry-out is fed back into the carry-in to achieve end-around-carry. To eliminate the unwanted race condition caused by the end-around-carry, one solution in the literature is to use an adder followed by an incrementer or a decrementer [7]. However, the carry propagation of the incrementer or decrementer slows down the operation of the modulo adders and subtractors especially for large word width such as 64-bits. In this section, we use the proposed binary increment and decrement technique to design fast and size-reduced modulo 2^n-1 adders and subtractors.

A. Proposed modulo adder

The residue sum of two *n*-bit residue digits, $(A + B)$ mod 2^n-1, is the residue of the sum $A + B$ with respect to the modulus 2^n-1. The operation may be defined as follows [7].

$$Z = |A+B|_{2^n-1} = \begin{cases} A+B, & A+B < 2^n - 1 \\ A+B-(2^n-1) = |A+B+1|_{2^n}, & A+B \ge 2^n - 1 \end{cases}$$

The modulo 2^n reduction is automatically performed if a *n*-bit adder is used. Note that the value "11⋯1" never occurs and that only one single representation "00⋯0" of zero exists. The modulo operation can be rewritten using the condition $A + B \ge 2^n$ as follows [7].

$$Z = |A+B|_{2^n-1} = \begin{cases} A+B, & A+B < 2^n \\ A+B-(2^n-1) = |A+B+1|_{2^n}, & A+B \ge 2^n \end{cases}$$

Here, zero has a double representation ("00⋯0" and "11⋯1"). Since the new condition $\ge 2^n$ equals to C_{out} =1, where C_{out} is the carry-out of the addition $A + B$, we have

$$Z = |A+B|_{2^n-1} = |A+B+C_{out}|_{2^n} \quad (3)$$

First, we use equation (4) to simplify the calculation of $|A+B|_{2^n-1}$ in equation (3).

Definition 1: Given any two *n*-bit binary input A and B, we define the addition operation as $A+B = 2^nC_{out} + S$. S, represented by $S_0\ S_1 \cdots S_{n-1}$, is the *n*-bit result of the addition $A+B$, and C_{out} is the carry-out bit.

Using Definition 1, we can transform equation (3) to equation (4) as follows.

$$Z = |A+B|_{2^n-1} = \begin{cases} S, & A+B < 2^n \\ S+1, & A+B \ge 2^n \end{cases} \quad (4)$$

where S is defined in Definition 1.

Equation (4) justifies the solution in the literature for the modulo 2^n-1 addition by using an adder followed by an incrementer. Now we can make use of the proposed MUX-based incrementer technique to design an efficient modulo 2^n-1 adder.

Proposition 4 Given any two n-bit binary input A and B, we get its modulo addition result $Z = |A+B|_{2^n-1}$ as follows.

$$Z = |A+B|_{2^n-1} \Leftrightarrow \begin{cases} Z_j = S_j \oplus (C_{out} S_0 S_1 \cdots S_{j-1}), & 1 \le j \le n-1 \\ Z_j = S_0 \oplus C_{out}, & j = 0 \end{cases} \quad (5)$$

(Proof is omitted due to lack of space).

Proposition 4 provides a novel way to design size-reduced, fast and low-power modulo circuits for modulo 2^n-1 addition due to its simplicity and modularity. As shown in Fig. 2(b), the new modulo adder is composed of a n-bit 2's complement adder, a SM and a data-out MUX array. The n-bit integer adder, which can be any other kind of 2's complement adder structure besides the CLA, is used to get the sum $S_0, S_1, \cdots, S_{n-1}$, and the carry-out C_{out}. The detail design of the SM is depicted in Fig. 2(a). The output of the SM, $D_0\ D_1 \cdots D_{n-1}$, form the selecting signals of the data-select MUX array which is used to select the correct output between the two cases in equation (5) respectively.

B. Proposed modulo subtractor

The operation of $(A - B)$ mod 2^n-1 may be defined as follows.

$$Z = |A-B|_{2^n-1} = \begin{cases} A-B, & 0 \le A-B < 2^n \\ A-B+2^n-1, & -2^n < A-B < 0 \end{cases}$$

Here zero has a double representation ("00$\cdots$0" and "11$\cdots$1"). We use equation (6) to simplify the calculation of $|A-B|_{2^n-1}$.

Definition 2: Given any two n-bit binary input A and B, we define the 2's complement subtraction operation as $A + \overline{B} + 1 = 2^n C_{out} + K$. K, represented by $K_0\ K_1 \cdots K_{n-1}$, is the n-bit result of the subtraction $A-B$, and C_{out} is the carry-out bit.

Using Definition 2, given any two n-bit binary input A and B, we get its modulo subtraction result $Z = |A-B|_{2^n-1}$ as follows.

$$Z = |A-B|_{2^n-1} = \begin{cases} K, & C_{out} = 1 \\ K-1, & C_{out} = 0 \end{cases} \quad (6)$$

where K is defined in Definition 2.

Equation (6) provides a solution for the modulo 2^n-1 subtraction by using an adder followed by a decrementer. Thus we can make use of the proposed MUX-based decrementer technique to design an efficient modulo 2^n-1 subtractor.

Proposition 5 Given any two n-bit binary input A and B, we get the modulo subtraction result $Z = |A-B|_{2^n-1}$ as follows

$$Z = |A-B|_{2^n-1} \Leftrightarrow \begin{cases} Z_j = K_j \oplus (\overline{C_{out}}\ \ \overline{K_0}\ \ \overline{K_1} \cdots \overline{K_{j-1}}), & 1 \le j \le n-1 \\ Z_j = \overline{K_0 \oplus C_{out}}, & j = 0 \end{cases}$$

(Proof is omitted due to lack of space).

Based on Proposition 5, we can design a new area-time efficient modulo 2^n-1 subtractor which has a similar structure to the modulo adder as shown in Fig. 2(b).

IV. COMPARISON RESULTS

Detailed comparison of the related modulo 2^n–1 adders are summarized in Table I, where the data for [8] are from [8, Table I]. In Table I, we have also included the results of [7, Table III] for an integer adder with a Sklansky prefix structure.

The estimation model used in Table I was originally presented in [9], which takes a two-input AND gate as one elementary gate for both area and delay. A XOR gate counts for two elementary gates. A MUX counts for one elementary gate. This model ignores the implementation characteristics such as fan-in, fan-out and the complexity of wiring. The validation of these parameters will be carried out below by FPGA implementation. The modulo adder proposed in [5] with the two level n-bit CLAs is shown in Fig. 2(c). For the convenience of comparison in Table I, we replace CLAs with two Sklansky parallel-prefix integer adders. For the same reason, we use the Sklansky prefix structure for the 2's complement adder in the proposed design.

As far as the area is concerned, Table I reveals that the proposed design requires the least implementation area for a modulo 2^n-1 addition. Although the parallel-prefix design in [8] is the fastest, it has complex interconnection and requires almost the same area as the two level adder design in [5]. Both of them have a double size of a n-bit integer adder. This makes the parallel-prefix design in [8] very expensive.

In addition to area, it can be noted that the proposed design is more versatile than the design in [8], since its n-bit 2's complement adder can be implemented with many kind of adder structure. This property makes the proposed design

TABLE I
PERFORMANCE COMPARISON OF MODULO ADDER

Adder	Area	Delay
Integer [7]	$\frac{3}{2} n \log n + 4n$	$2 \log n + 3$
[5]	$3n \log n + 9n$	$4 \log n + 8$
[8]	$3n \log n + 4n$	$2 \log n + 3$
Proposed	$2n \log n + 5n$	$3 \log n + 4$

TABLE II
IMPLEMENTATION AND COMPARISON

	16-bit			64-bit		
	Power (mw)	Cell Area	Time (ns)	Power (mw)	Cell Area	Time (ns)
C_1 : in [5]	13.8	762	22.7	56.1	3300	35.1
C_2 : Proposed	11.8	534	20.2	50.8	2262	32.2
$\frac{C_1 - C_2}{C_1} \times 100\%$	14.5	29.9	11	9.4	31.5	8.3

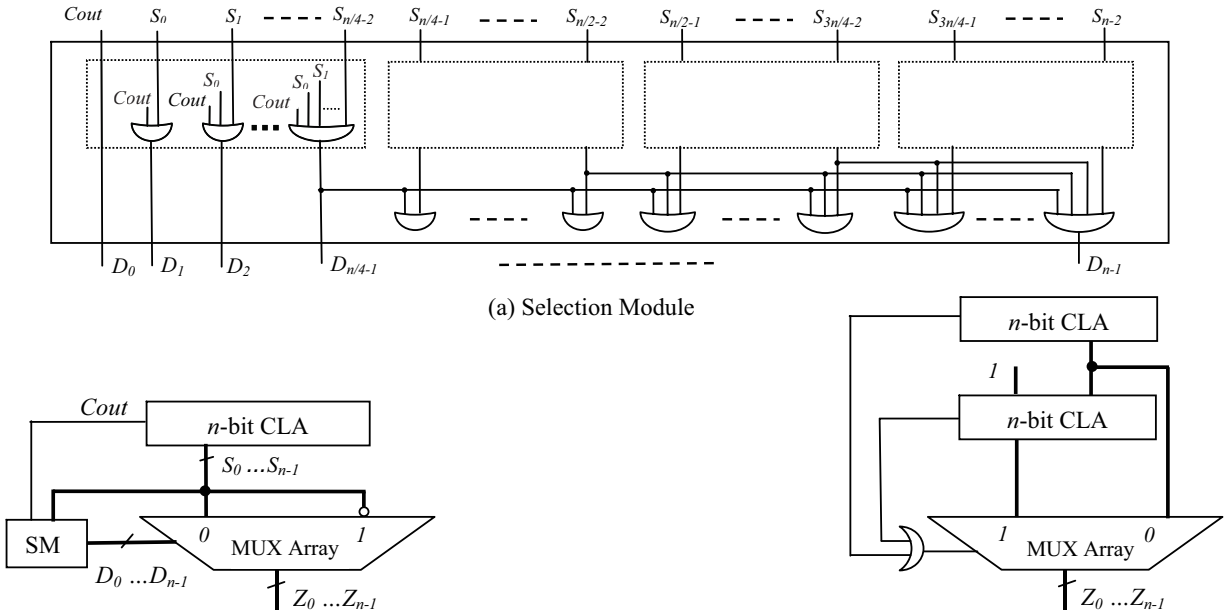

Fig. 2. *n*-bit modulo 2^n–1 adders.

suitable for more general purpose. For example, if the target implementation technology is FPGA, then we can use CLA or carry-propagation adder (CPA) since each slice of FPGA includes dedicated ripple carry logic. If the design is mapped to CMOS technology, then we may prefer the parallel-prefix adders since the interconnection of CMOS is more efficient than that of FPGA.

To get a practical performance measure of the modulo 2^n–1 adders, the proposed adder and the design in [5] based on *n*-bit CLAs are implemented using Xilinx FPGA technology for the 16-bit and 64-bit cases. The parallel-prefix design has complex interconnection and is not suitable for FPGA implementation. The comparison results of Table I show that the proposed design requires less hardware resources than the parallel-prefix design in [8] while providing a comparable operation speed. The synthesis and implementation tools are Xillinx Synthesis Tool (XST) and Xilinx Integrated Software Environment (ISE) 6.3i. The power analysis tool is XPower. The target technology is a Xillinx Virtex 2 xc2v2000ff896-4 FPGA. The performance evaluation is carried out in terms of power, area and delay. To get the power consumption of the internal core circuit, we use the default VCCInt and Ambient Temp values of 1.5V and 25 degrees Celsius. All the inputs are clocked at 20 MHz. The reported area is evaluated using equivalent gate count. The results of FPGA implementation in Table II show that the proposed adder consumes almost 30% less hardware than the design in [5]. And the delay and the power are also reduced around 10%.

V. CONCLUSION

In this paper, we have presented a versatile area-reduced scheme for modulo 2^n–1 adders and subtractors using a novel MUX-based increment/decrement algorithm. A FPGA based comparison of the proposed modulo adder and the conventional modulo adders has been carried out. The results show that the proposed adder reduces the area close to 30% with compared to the previous design in [5]. And the delay and the power are also reduced around 10%. In addition, it has been shown that the proposed design requires less hardware resources than the parallel-prefix modulo adders in [8] while providing a comparable operation speed.

REFERENCE

[1] M. A. Soderstrand, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. New York: IEEE Press, 1986.

[2] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," *Advances in Cryptology – EUROCRYPT'90*, Berlin, Germany: Springer-Verlag, pp. 389-404, 1990.

[3] B. J. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison Wesley Publishing Company, 1989.

[4] J. J. Shedletsky, "Comment on the sequential and indeterminate behavior of an end-around-carry adder," *IEEE Trans. Computers*, vol. C-26, pp. 271-272, Mar. 1977.

[5] M. A. Bayoumi, G. A. Jullien, and W. C. Miller. "A VLSI implementation of residue adders," *IEEE Trans. Circuits and Systems*, vol. CAS-34, pp. 284-288, Mar. 1987.

[6] C. Efstathiou, D. Nikolos and J. Kalamatianos, "Area-time efficient modulo 2^n-1 adder design," *IEEE Trans. Circuits and Systems-II*, vol. 41, No. 7, July 1994.

[7] R. Zimmermann, "Efficient VLSI implementation of modulo (2^n±1) addition and multiplication," *Proc. 14th IEEE Symposium on Computer Arithmetic*, pp. 158-167, 1999.

[8] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-speed parallel-prefix modulo 2^n-1 adders," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 673-680, Jul. 2000.

[9] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1,163-1,170, Oct. 1993.