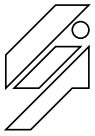
More on Modulo 2n -1 Addition

Jean-Luc Beuchat



Laboratoire de l'Informatique du Parallélisme

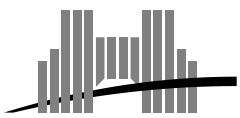
École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

More on Modulo $2^n - 1$ *Addition*

Jean-Luc Beuchat

February 2003

Research Report N° 2003-14



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

More on Modulo $2^n - 1$ Addition

Jean-Luc Beuchat

February 2003

Abstract

This brief paper describes an improvement of the FPGA implementation of the modulo $(2^n - 1)$ adder investigated in a previous research report.

Keywords: Computer arithmetic, modulo $2^n - 1$ addition, FPGA

Résumé

Ce petit article décrit une amélioration de l'implantation FPGA de l'additionneur modulo $(2^n - 1)$ décrit dans un précédent rapport de recherche.

Mots-clés: Arithmétique des ordinateurs, addition modulo $2^n - 1$, FPGA

1 Introduction

This brief report describes an improvement of the FPGA implementation of the modulo (2^n-1) adder investigated in [1]. Modulo (2^n-1) addition, or one's complement addition, is defined by [2]:

$$(x+y) \bmod (2^n-1) = \begin{cases} (x+y+1) \bmod 2^n & \text{if } x+y+1 \geq 2^n, \\ (x+y) \bmod 2^n & \text{if } x+y+1 < 2^n. \end{cases} \quad (1)$$

Figure 1a depicts the architecture of the corresponding hardware operator. Due to the condition $x+y+1 \geq 2^n$, we perform two additions in parallel and select the correct result with a multiplexer. Remember that zero has a double representation in one's complement, namely "$0\ldots0$" and "$1\ldots1$" (i.e. 0 is congruent to 2^n-1 (modulo 2^n-1)). If the computation path accommodates the second encoding of zero, Equation (1) can be rewritten as follows:

$$(x+y) \bmod (2^n-1) = \begin{cases} (x+y+1) \bmod 2^n & \text{if } x+y \geq 2^n, \\ (x+y) \bmod 2^n & \text{if } x+y < 2^n. \end{cases} \quad (2)$$

Note that the carry-out c_{out} from the sum $x+y$ indicates whether the incrementation must be performed. It is still possible to evaluate $x+y$ and $x+y+1$ in parallel, and to choose the correct result according to c_{out} (Figure 1b). An alternate architecture, illustrated on Figure 1c, simply adds c_{out} to the $x+y$. Therefore, the double representation of zero allows the removal of the multiplexer and leads to a smaller circuit.

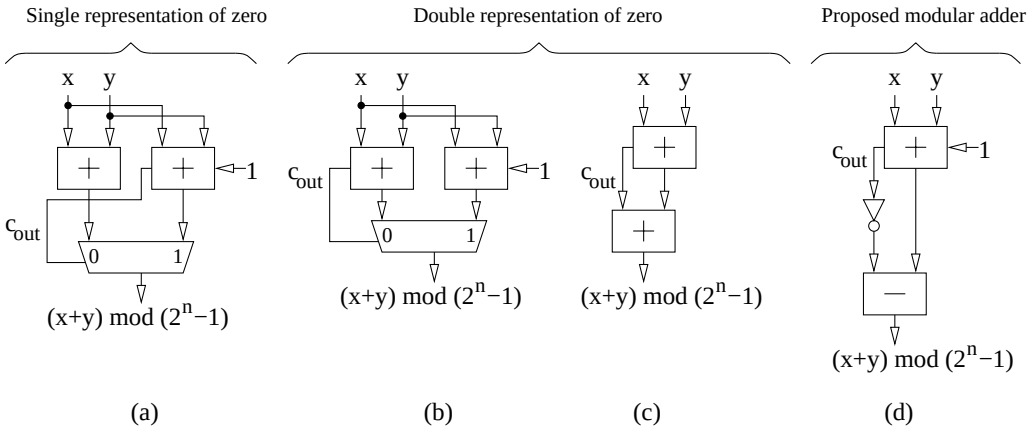


Figure 1: Four modulo (2^n-1) adders.

2 A New Modulo (2^n-1) Adder

We propose here a new modulo (2^n-1) addition algorithm that avoids the double representation of zero, while only requiring two adders. The algorithm is defined as follows:

$$(x+y) \bmod (2^n-1) = ((x+y+1) \bmod 2^n - \underbrace{(1-(x+y+1) \operatorname{div} 2^n)}_{\overline{c}_{out}}) \bmod 2^n. \quad (3)$$

Let us show that this algorithm is equivalent to the modulo (2^n+1) addition scheme (1). It suffices to consider two cases:

1. For $x+y+1 < 2^n$, we have:

$$\begin{aligned} (x+y) \bmod (2^n-1) &= (\underbrace{(x+y+1) \bmod 2^n}_{=x+y+1} - (1 - \underbrace{(x+y+1) \operatorname{div} 2^n}_{=0})) \bmod 2^n \\ &= (x+y) \bmod 2^n. \end{aligned}$$

2. For $x+y+1 \geq 2^n$, we obtain:

$$\begin{aligned} (x+y) \bmod (2^n-1) &= ((x+y+1) \bmod 2^n - (1 - \underbrace{(x+y+1) \operatorname{div} 2^n}_{=1})) \bmod 2^n \\ &= ((x+y+1) \bmod 2^n) \bmod 2^n = (x+y+1) \bmod 2^n. \end{aligned}$$

Figure 1d illustrates the architecture of this new operator which requires two carry-propagate adders and an inverter. On Virtex-E or Virtex-II devices, this inverter is implemented within the carry chain and the modulo $(2^n - 1)$ adder fits into a single CLB column. We have then carried out a series of experiments in order to compare the four architectures described in this paper[1] (Table 1). These results indicate that our new operator requires actually the same area as the modulo $(2^n - 1)$ adder depicted on Figure 1c.

Table 1: Comparison of the four modulo $(2^n - 1)$ adders on a XCV50E-6 device.

	$n = 4$	$n = 8$	$n = 12$	$n = 16$	$n = 20$	$n = 24$	$n = 28$	$n = 32$
Fig. 1a	6 slices 8.1 ns	12 slices 9.6 ns	18 slices 10.5 ns	24 slices 11.2 ns	30 slices 12.4 ns	36 slices 12.8 ns	42 slices 13.7 ns	48 slices 14.9 ns
Fig. 1b	6 slices 8.2 ns	12 slices 9.5 ns	18 slices 10.7 ns	24 slices 11.2 ns	30 slices 12.6 ns	36 slices 13.5 ns	42 slices 13.7 ns	48 slices 14.9 ns
Fig. 1c	4 slices 9.1 ns	8 slices 9.9 ns	12 slices 11.7 ns	16 slices 12.5 ns	20 slices 13.8 ns	24 slices 16.1 ns	28 slices 16.3 ns	32 slices 18.1 ns
Fig. 1d	4 slices 8.7 ns	8 slices 10.8 ns	12 slices 11.4 ns	16 slices 12.4 ns	20 slices 13.4 ns	24 slices 16.1 ns	28 slices 15.8 ns	32 slices 16.3 ns

References

[1] Jean-Luc Beuchat. Some Modular Adders and Multipliers for Field Programmable Gate Arrays. Technical Report 2002-37, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, October 2002.

[2] R. Zimmermann. Efficient VLSI Implementation of Modulo $(2^n \pm 1)$ Addition and Multiplication. In *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pages 158–167, Adelaide, Australia, April 1999.

[1] The VHDL code was synthesized and implemented on a XCV50E-6 device with ISE 5.1.03i.