More on Modulo 2n -1 Addition
Jean-Luc Beuchat

To cite this version:
Jean-Luc Beuchat. More on Modulo 2n -1 Addition. [Research Report] LIP RR-2003-14, Laboratoire de l’informatique du parallélisme. 2003, 2+2p. hal-02102102

HAL Id: hal-02102102
https://hal-lara.archives-ouvertes.fr/hal-02102102
Submitted on 17 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
More on Modulo $2^n - 1$ Addition

Jean-Luc Beuchat

February 2003

Research Report N° 2003-14
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]:

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

(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...0” and “1...1” (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:

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

(2)

Note that the carry-out \(c_{\text{out}}\) 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 \(c_{\text{out}}\) (Figure 1b). An alternate architecture, illustrated on Figure 1c, simply adds \(c_{\text{out}}\) 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.](image)

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:

\[
(x + y) \mod (2^n - 1) = ((x + y + 1) \mod 2^n - (1 - (x + y + 1) \text{ div } 2^n)) \mod 2^n. 
\]  

(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:

\[
(x + y) \mod (2^n - 1) = ((x + y + 1) \mod 2^n - (1 - (x + y + 1) \text{ div } 2^n)) \mod 2^n \\
= (x + y) \mod 2^n.
\]

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

\[
(x + y) \mod (2^n - 1) = ((x + y + 1) \mod 2^n - (1 - (x + y + 1) \text{ div } 2^n)) \mod 2^n \\
= ((x + y + 1) \mod 2^n) \mod 2^n = (x + y + 1) \mod 2^n.
\]
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        | 12 slices       | 18 slices       | 24 slices       | 30 slices       | 36 slices       | 42 slices       | 48 slices       |
|                 | 8.1 ns          | 9.6 ns          | 10.5 ns         | 11.2 ns         | 12.4 ns         | 12.8 ns         | 13.7 ns         | 14.9 ns         |
| Fig. 1b         | 6 slices        | 12 slices       | 18 slices       | 24 slices       | 30 slices       | 36 slices       | 42 slices       | 48 slices       |
|                 | 8.2 ns          | 9.5 ns          | 10.7 ns         | 11.2 ns         | 12.6 ns         | 13.5 ns         | 13.7 ns         | 14.9 ns         |
| Fig. 1c         | 4 slices        | 8 slices        | 12 slices       | 16 slices       | 20 slices       | 24 slices       | 28 slices       | 32 slices       |
|                 | 9.1 ns          | 9.9 ns          | 11.7 ns         | 12.5 ns         | 13.8 ns         | 16.1 ns         | 16.3 ns         | 18.1 ns         |
| Fig. 1d         | 4 slices        | 8 slices        | 12 slices       | 16 slices       | 20 slices       | 24 slices       | 28 slices       | 32 slices       |
|                 | 8.7 ns          | 10.8 ns         | 11.4 ns         | 12.4 ns         | 13.4 ns         | 16.1 ns         | 15.8 ns         | 16.3 ns         |

References


---

\(^1\)The VHDL code was synthesized and implemented on a XCV50E-6 device with ISE 5.1.03i.