

Comment on “Fast Parallel Prefix Modulo 2^n+1 Adder”

Ghassem Jaberipur and Hanieh Alavi, ECE dept., Shahid Beheshti University

Abstract— Costas Efstathiou et al present (IEEE Trans. Computers, Vol. 53, No. 9 pp. 1211-1216) an n -bit totally parallel prefix (TPP) implementation of modulo 2^n+1 adders with $6 + 2 \log n$ latency in terms of unit gate delay. We locate a flaw in the logic equation for the most significant bit and present a simple counter example to prove this claim. We provide the relevant correct equation and its derivation details. We also show that it can be implemented within the TPP tree, without additional latency. Furthermore, despite the correctness of the equations for individual carry signals, we point out a missing parallel prefix operand in the corresponding general equation. In lack of any derivation or proof for the latter, we provide the relevant correct equation with derivation details.

Index Terms— Binary adders, Modulo 2^n+1 arithmetic, Parallel prefix adders, RNS.

1 INTRODUCTION

THE moduli set $\{2^n - 1, 2^n, 2^n+1\}$ is popular in the applications of residue number system (RNS).

Modular adders for the 2^n and 2^n-1 channels have been reported via n -bit parallel prefix and n -bit totally parallel prefix (TPP) trees, with $(3 + 2 \log n)$ latency in terms of unit gate delay [2]. Timing coordination within the three RNS channels calls for modulo 2^n+1 adders with $O(\log n)$ latency. This has motivated the authors of [1] to design such adders via a $(\log n)$ level TPP, where the overall latency is $6 + 2 \log n$. Unfortunately however, there is a flaw in the logical equation for the most significant bit (MSB) of the sum that leads to wrong results in several instances of the input operands.

In this comment we underline the aforementioned flaw via a counter example, provide the derivation details of the pertinent correct equation and its implementation within the same $(\log n)$ level TPP tree of [1] such that the $(6 + 2 \log n)$ latency is preserved. Moreover, we offer the derivation details for Equation (4) of [1], where our motive is twofold: First, this equation has a key role since it computes all the TPP carries, while no derivation or proof is given for it in [1]. Secondly, there is a missing parallel prefix operand, although all applications of this equation, for $n = 8$, are correct.

2 THE FLAW

The modulo 2^n+1 addition scheme in [1] primarily computes $M = A + B + 2^n - 1$, where A and B are the $n+1$ -bit operands in $[0, 2^n]$. Equation (1), adapted from [1], defines $R = r_n r_{n-1} \dots r_1 r_0 = |A + B|_{2^n+1}$ in terms of M , where $|X|_m$ stands for X modulo m and \bar{x} stands for the complement of x .

$$R = |m_n m_{n-1} \dots m_1 m_0 + (2^n + 1) \overline{m_{n+1}}|_{2^n+1} \quad (1)$$

Fig. 1 depicts a typical computation of $M (= S+C)$, where $s_i = a_i \oplus b_i$, $c_i = a_i \vee b_i$, $s_n = a_n \oplus b_n$ and $c_n = a_n \wedge b_n$.

A	a_n	a_{n-1}	\dots	a_1	a_0	
B	b_n	b_{n-1}	\dots	b_1	b_0	
2^n-1	0	1	\dots	1	1	
S	s_n	s_{n-1}	\dots	s_1	s_0	
C	c_n	c_{n-1}	\dots	c_1	c_0	
M	m_{n+1}	m_n	m_{n-1}	\dots	m_1	m_0

Fig. 1: Computation of M

The computation of R can be analyzed as follows:

r_0 : $r_0 = m_0 \oplus \overline{m_{n+1}} = s_0 \oplus \overline{c_n \vee G_{n,1}}$, where $G_{n,1}$ is the carry-out of position n in the original computation of M . Note that since $M \leq 2^{n+1}+2^n - 1$ no carry is generated in the computation of $c_n + G_{n,1}$. Therefore, $m_{n+1} = c_n \vee G_{n,1}$.

r_i : ($1 \leq i \leq n-1$): $r_i = m_i \oplus G_{i-1,1}^* = s_i \oplus c_{i-1} \oplus G_{i-1,1}^*$, where $G_{i-1,1}^*$ is the carry into position i within the addition $m_{n-1} \dots m_0 + c_n \vee G_{n,1}$.

r_n : $r_n = s_n + c_{n-1} + \overline{m_{n+1}} + G_{n-1,1}^*$. This is simply sum of the bits in position n of Fig. 1.

The $S+C$ stage of Fig. 1 is trusted to a TPP tree, with $c_{in} = c_n \vee G_{n,1}$. However, there is flaw in the actual equation implemented in Fig. 3 of [1] for r_n .

2.1 The flaw in the computation of r_n

It is rightly stated in [1] that $r_n = 1$ if $A+B = 2^n$. Then the authors, without providing any proof, conclude that $r_n = \overline{c_n} \wedge P_n \wedge s_0$, where P_n is the group propagate signal from position 1 to n . It is not difficult to prove that $(A + B = 2^n)$ implies $(\overline{c_n} \wedge P_n \wedge s_0 = 1)$. However, the converse (i.e., $(\overline{c_n} \wedge P_n \wedge s_0 = 1) \Rightarrow (A + B = 2^n)$) is not always true. In fact it fails in 25% of the cases of input data for $n = 8$. This claim is supported by exhaustive test via VHDL simulation. However, Example 1 below clearly demonstrates the flaw.

Example 1 (Counter example for $r_n = \overline{c_n} \wedge P_n \wedge s_0$): Consider an instance of Fig. 1, for $n = 4$, where $c_4 = 0$, $s_0 = 1$, and $P_4 = 1$ lead to $\overline{c_n} \wedge P_n \wedge s_0 = 1$. However, $R = |12+12|_{17}=7$, which leads to $r_4 = 0$. ◀

$A = 12$	0	1	1	0	0
$B = 12$	0	1	1	0	0
$2^4-1 = 15$	0	1	1	1	1
S	0	1	1	1	1
C	0	1	1	0	0
M	1	0	0	1	1

Fig. 2: An instance of Fig. 1 for $n = 4$

The correct equation, as will be derived in Section 3, is:

$$r_n = (c_n \vee s_n \wedge c_{n-1} \vee (s_n \vee c_{n-1}) \wedge \overline{G_{n-1,1}}) \oplus \overline{G_{n-1,1}} \quad (3)$$

The correctness of the latter is exhaustively tested via VHDL simulation for $n = 8$. Besides the latter flaw, a parallel prefix term $(c_n \vee g_n, p_n)$ is missing in the third part of Equation (4) in [1]. The corrected Equation (4), as will be derived in the next section, is as follows, where

$$\varphi(G, P) = G: \quad G_{i,1}^* = \varphi \left((G_{i,1}, P_{i,1}) \circ s_0 \wedge (c_n \vee g_n, p_n) \circ (G_{n-1,i+1}, P_{n-1,i+1}) \right) \quad (4)$$

3 THE CORRECTED DESIGN

The corrected Equations (3) and (4) are derived below.

3.1 Derivation of the most significant bit r_n

Recalling the arithmetic equation for r_n from Section 2 (i.e., $r_n = s_n + c_{n-1} + \overline{m_{n+1}} + G_{n-1,1}^*$), and $m_{n+1} = G_{n,1} \vee c_n$, the MSB r_n can be computed by the logical Equation (5).

$$r_n = s_n \oplus c_{n-1} \oplus \overline{m_{n+1}} \oplus G_{n-1,1}^* = ((s_n \oplus (G_{n,1} \vee c_n)) \oplus c_{n-1}) \oplus G_{n-1,1}^* \quad (5)$$

Given that $s_n \wedge c_n = 0$, $c_n \wedge \overline{s_n} = c_n$, $G_{n,1} = g_n \vee p_n \wedge G_{n-1,1}$, $g_n = s_n \wedge c_{n-1}$, and $p_n = s_n \vee c_{n-1}$, the inner parenthesized XOR equation can be simplified as follows:

$$\begin{aligned} s_n \oplus (G_{n,1} \vee c_n) &= (s_n \vee c_n \vee G_{n,1}) \wedge \overline{s_n \wedge G_{n,1} \vee s_n \wedge c_n} \\ &= s_n \wedge \overline{G_{n,1}} \vee \overline{s_n} \wedge G_{n,1} \vee c_n \wedge \overline{s_n} \vee c_n \wedge G_{n,1} \\ &= c_n \vee s_n \wedge \overline{G_{n,1}} \vee \overline{s_n} \wedge G_{n,1} = c_n \vee s_n \wedge g_n \vee p_n \wedge G_{n-1,1} \vee \overline{s_n} \\ &\wedge (g_n \vee p_n \wedge G_{n-1,1}) = c_n \vee s_n \wedge \overline{g_n} \wedge (\overline{p_n} \vee G_{n-1,1}) \vee \overline{s_n} \wedge \\ &s_n \wedge c_{n-1} \vee \overline{s_n} \wedge (s_n \vee c_{n-1}) \wedge G_{n-1,1} \\ &= c_n \vee s_n \wedge (\overline{s_n} \vee \overline{c_{n-1}}) \wedge (\overline{s_n} \wedge \overline{c_{n-1}} \vee G_{n-1,1}) \vee \overline{s_n} \wedge c_{n-1} \\ &\wedge G_{n-1,1} = c_n \vee s_n \wedge \overline{c_{n-1}} \wedge \overline{G_{n-1,1}} \vee \overline{s_n} \wedge c_{n-1} \wedge G_{n-1,1} \end{aligned}$$

Given that $c_n = 1 \Rightarrow c_{n-1} = 0 \Rightarrow c_n \wedge \overline{c_{n-1}} = c_n$, the outer parenthesized XOR equation within Equation (5) can now be simplified as follows:

$$\begin{aligned} (c_n \vee s_n \wedge \overline{c_{n-1}} \wedge \overline{G_{n-1,1}} \vee \overline{s_n} \wedge c_{n-1} \wedge G_{n-1,1}) \oplus c_{n-1} \\ &= (c_n \vee s_n \wedge \overline{c_{n-1}} \wedge \overline{G_{n-1,1}} \vee \overline{s_n} \wedge c_{n-1} \wedge G_{n-1,1} \vee c_{n-1}) \\ &\wedge (c_n \vee s_n \wedge \overline{c_{n-1}} \wedge \overline{G_{n-1,1}} \vee \overline{s_n} \wedge c_{n-1} \wedge G_{n-1,1}) \wedge c_{n-1} \\ &= (c_n \vee s_n \wedge \overline{c_{n-1}} \vee c_{n-1}) \wedge (c_n \wedge c_{n-1} \vee \overline{s_n} \wedge c_{n-1} \wedge G_{n-1,1}) \\ &= c_n \wedge s_n \vee c_n \wedge \overline{c_{n-1}} \vee c_n \wedge \overline{G_{n-1,1}} \vee s_n \wedge \overline{G_{n-1,1}} \vee s_n \wedge c_{n-1} \\ &\vee c_{n-1} \wedge \overline{G_{n-1,1}} = c_n \vee s_n \wedge c_{n-1} \vee (s_n \vee c_{n-1}) \wedge \overline{G_{n-1,1}} \end{aligned}$$

It only remains to apply the latter into Equation (5) that leads to the desired Equation (3). Direct implementation of this equation leads to the overall latency of $7 + 2 \log n$.

3.2 Derivation of $G_{i,1}^*$ ($1 \leq i \leq n-2$)

$$\begin{aligned} G_{i,1}^* &= G_{i,1} \vee P_{i,1} \wedge G_0^* = G_{i,1} \vee P_{i,1} \wedge (s_0 \wedge c_n \vee g_n \vee p_n G_{n-1,1}) \\ &= G_{i,1} \vee P_{i,1} \wedge s_0 \wedge c_n \vee g_n \wedge (\overline{p_n} \vee G_{n-1,i+1} \vee P_{n-1,i+1} \wedge G_{i,1}) \\ &= G_{i,1} \vee (P_{i,1} \wedge s_0 \wedge \overline{c_n} \wedge \overline{g_n} \wedge \overline{p_n}) \vee (P_{i,1} \wedge s_0 \wedge c_n \vee g_n \\ &\wedge \overline{G_{n-1,i+1}} \wedge \overline{P_{n-1,i+1}}) \vee (P_{i,1} \wedge s_0 \wedge c_n \vee g_n \wedge \overline{G_{n-1,i+1}} \wedge G_{i,1}) \\ &= G_{i,1} \vee (P_{i,1} \wedge s_0 \wedge \overline{c_n} \wedge \overline{p_n}) \vee P_{i,1} \wedge s_0 \wedge c_n \vee g_n \wedge \overline{G_{n-1,i+1}} \\ &= G_{i,1} \vee P_{i,1} \wedge s_0 \wedge (\overline{c_n} \vee \overline{p_n} \vee c_n \vee g_n \wedge \overline{G_{n-1,i+1}}) \\ &= G_{i,1} \vee P_{i,1} \wedge s_0 \wedge (\overline{c_n} \vee p_n)(c_n \vee g_n \vee G_{n-1,i+1}) \\ &= G_{i,1} \vee P_{i,1} \wedge s_0 \wedge c_n \vee g_n \vee p_n \wedge G_{n-1,i+1} \\ &= \mathcal{G}((G_{i,1}, P_{i,1}) \circ s_0 \wedge (c_n \vee g_n, p_n) \circ (G_{n-1,i+1}, P_{n-1,i+1})) \end{aligned}$$

Given the above carry signals, which are computable via a TPP tree exactly as in [1], the final sum bits r_i can be computed as follows:

$$r_i = h_i \oplus G_{i-1,1}^* = s_i \oplus c_{i-1} \oplus G_{i-1,1}^* \quad (1 \leq i \leq n-1)$$

3.3 Implementation of r_n within the TPP tree

The left operand of the XOR in Equation (3) can be represented as a prefix equation and implemented within the TPP tree as in Equation (6) and Fig. 3, where $\gamma = a_n \vee b_n \vee c_{n-1}$ and $\pi = a_n \wedge b_n \vee (a_n \vee b_n) \wedge c_{n-1}$.

The γ and π equations are justified as follows:

$$\begin{aligned} c_n \vee s_n \wedge c_{n-1} \vee (s_n \vee c_{n-1}) \wedge \overline{G_{n-1,1}} \\ &= \overline{a_n \wedge b_n \vee (a_n \vee b_n) \wedge c_{n-1}} \wedge \overline{a_n \wedge b_n \wedge c_{n-1} \vee (a_n \vee b_n) \wedge a_n \wedge b_n} \\ &\wedge \overline{G_{n-1,1} \vee c_{n-1} \wedge G_{n-1,1}} \\ &= \overline{a_n \wedge b_n \vee (a_n \vee b_n) \wedge c_{n-1} \vee (a_n \vee b_n \vee c_{n-1}) \wedge \overline{G_{n-1,1}}} \\ &= \overline{a_n \vee b_n \vee c_{n-1} \vee a_n \wedge b_n \vee (a_n \vee b_n) \wedge c_{n-1} \wedge G_{n-1,1}} \\ &= \mathcal{G}((\gamma, \pi) \circ (G_{n-1,1}, P_{n-1,1})) \end{aligned}$$

$$r_n = \mathcal{G}((\gamma, \pi) \circ (G_{n-1,1}, P_{n-1,1})) \oplus G_{n-1,1}^* \quad (6)$$

The prefix node that could compute $G_{n-1,1}^*$ in the last row and column $n-1$ of the TPP tree is missing in Fig. 3 of [1]. This and the path leading to r_8 , defined by Equation (7), are now added in the new Fig. 3 below.

$$r_8 = \mathcal{G}((\gamma, \pi) \circ (g_7, p_7) \circ \dots (g_1, p_1)) \oplus G_{7,1}^* \quad (7)$$

The latency of γ and π is 2 and 3 unit gate delay. Therefore, the prefix operand (γ, π) is ready as soon as all other prefix operands are. This leads to availability of r_n at time $5 + 2 \log n$ in terms of unit gate delay.

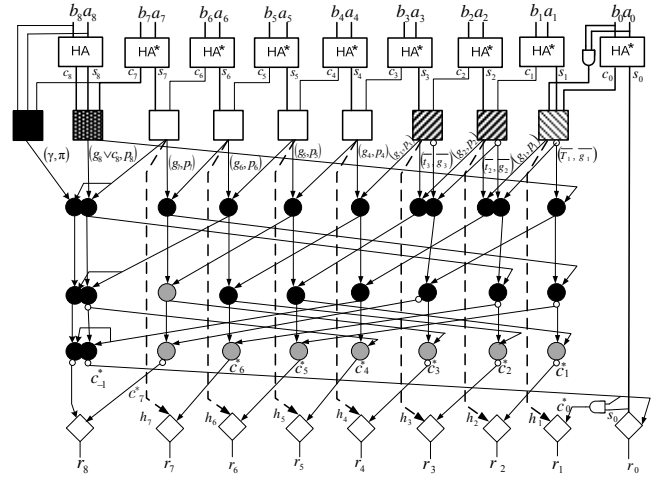


Fig. 3: The corrected modulo 2^n+1 TPP adder

REFERENCES

- [1] Efstathiou C., H. T. Vergos, and D. Nikolos, "Fast Parallel-Prefix 2^n+1 Adder", IEEE Trans. on Computers, Vol. 53, No. 9, pp. 1211-1216, September 2004.
- [2] Kalamatianos, "High-Speed Parallel-Prefix Modulo 2^n-1 Adders," IEEE Trans. Computers, Vol. 49, No. 7, special issue on computer arithmetic, pp. 673-680, July 2000.
- [3] Vergos H.T. and C. Efstathiou, "Novel Modulo 2^n+1 Multipliers", Proc. of the 9th EUROMICRO Conference on Digital system Design, Dubrovnik, pp.168-175, 2006.