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

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

Abstract—Costas Efstatiou 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 derivation details with derivation details.

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

1 INTRODUCTION

The modulo 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-bit operands in [0, 2^n]. Equation (1), adapted from [1], defines R = r_n r_{n-1} ... 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.

\[ R = [m_n m_{n-1} ... m_0 + (2^n + 1)\bar{m}_{n+1}]_{2^{n+1}} \] (1)

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

\[
\begin{array}{cccccccc}
A & a_0 & a_{n-1} & ... & a_1 & a_0 \\
B & b_0 & b_{n-1} & ... & b_1 & b_0 \\
2^n-1 & 0 & 1 & 1 & 1 \\
S & s_0 & s_{n-1} & ... & s_1 & s_0 \\
C & c_0 & c_{n-1} & ... & c_1 & c_0 \\
M & m_0 & m_{n-1} & ... & m_1 & m_0 \\
\end{array}
\]

Fig. 1: Computation of M

The computation of R can be analyzed as follows:

\[ r_0 = m_0 \oplus \bar{m}_{n+1} = s_0 \oplus \bar{c}_n \lor G_{n,1} \]

\( G_{n,1} \) 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, \( m_{n+1} = c_n \lor G_{n,1} \).

\( r_i \) (1 \leq i \leq n-1): \( r_i = m_i \lor G_{i-1,1} = s_i \lor c_{i-1} \lor G_{i-1,1} \),

where \( G_{i-1,1} \) is the carry into position i within the addition \( m_{i-1} + ... + m_0 + c_n \lor G_{n,1} \).

\[ r_n = s_n + c_{n-1} + \bar{m}_{n+1} + G_{n-1,-1} \]

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 \( c_{im} = c_n \lor G_{im,1} \). 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 \( r_n = c_n \lor P_n \land s_0 \), 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 \( c_n \lor P_n \land s_0 = 1 \). However, the converse (i.e., \( c_n \lor P_n \land s_0 = 1 \) \( \Rightarrow \) \( A+B = 2^n \)) 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 \( r_n = c_n \lor P_n \land s_0 \)):

Consider an instance of Fig. 1, for n = 4, as depicted in Fig. 2, where \( c_i = 0, s_i = 1 \), and \( P_i = 1 \) lead to \( c_n \lor P_n \land s_0 = 1 \). However, R = 12 | 12 | 24 | 7, which leads to \( r_0 = 0 \).}

\[
\begin{array}{cccc}
A & 12 & 0 & 1 & 1 & 0 & 0 \\
B & 12 & 0 & 1 & 1 & 0 & 0 \\
2^n-1 & 0 & 1 & 1 & 1 & 1 \\
S & 0 & 1 & 1 & 1 & 1 \\
C & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

\[ M \]
\[ 1 \]
\[ 0 \]
\[ 0 \]
\[ 1 \]
\[ 1 \]
\[ 1 \]

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

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

\[ r_n = (c_n \lor s_0 \land c_{n-1} \lor (s_n \lor c_{n-1}) \lor G_{n-1,1}) \lor G_{n-1,-1} \] (3)

The correctness of the latter is exhaustively tested via VHDL simulation for n = 8. Besides the latter flaw, a parallel prefix term \( c_n \lor g_{n,P_n} \) 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 \( g(G, P) = G \):

\[ G_{i,1} = g \left( G_{i-1,1} \lor P_{i-1} \right) \lor s_0 \lor (c_n \lor g_{n,P_n} \lor (G_{n-1,1} \lor i \lor P_{n-1,1} \lor i)) \] (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., $r_n = \sum_0^n + m_{n+1} + G_{n-1,1}$), and $m_{n+1} = G_{n,1} \lor c_n$, the MSB $r_n$ can be computed by the logical equation (5).

$$r_n = S_n \oplus c_{n-1} \oplus m_{n+1} \oplus G_{n-1,1} = \left((S_n \oplus G_{n,1} \lor c_n) \oplus c_{n-1} \right) \oplus G_{n-1,1}$$ (5)

Given that $s_n \land c_n = 0$, $c_n \land \bar{s}_n = c_n$, $G_{n,1} \lor p_n \land G_{n-1,1}$, $g_n = s_n \land c_{n-1}$, and $p_n = c_n \land \bar{v}_n$, the inner parenthesized XOR equation can be simplified as follows:

$$s_n \oplus G_{n,1} \lor c_n = (s_n \lor c_n \lor G_{n,1}) \land s_n \lor G_{n-1,1} \land c_n$$

$$= c_n \lor s_n \land \bar{G}_{n-1,1} \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor c_n \lor s_n \land G_{n-1,1} \land c_n$$

$$= c_n \lor s_n \land \bar{G}_{n-1,1} \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= G_{n-1,1} = c_n \lor s_n \land \bar{G}_{n-1,1} = s_n \land c_n$$

$$= G_{n-1,1} = c_n \lor s_n \land \bar{G}_{n-1,1} = s_n \land c_n$$

$$\Rightarrow G_{n-1,1} = c_n \lor s_n \land \bar{G}_{n-1,1} = s_n \land c_n$$

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

$$c_n \lor s_n \land \bar{G}_{n-1,1} \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor c_n \lor s_n \land G_{n-1,1} \land c_n$$

$$= c_n \lor s_n \land \bar{G}_{n-1,1} \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor s_n \land \bar{G}_{n-1,1} \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

$$= c_n \lor \bar{s}_n \land \bar{c}_{n-1} \lor \bar{G}_{n-1,1} \lor s_n \land c_n$$

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 $G_i^{l+1}$ ($1 \leq i \leq n-2$)

$$G_i^{l+1} = G_{i-1} \lor P_{i-1} \land G_{i-1} = G_{i-1} \lor P_{i-1} \land (s_0 \land c_{n+1} \lor g_n \lor p_n \lor p_{n-1} \lor G_{n-1,1})$$

$$= G_{i-1} \lor P_{i-1} \land s_0 \land c_{n-1} \lor g_n \lor p_n \lor p_{n-1} \lor G_{n-1,1}$$

$$= G_{i-1} \lor P_{i-1} \land s_0 \land c_{n-1} \lor g_n \lor p_n \lor p_{n-1} \lor G_{n-1,1}$$

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:

$$r_i = h_i \oplus G_{i-1} = s_i \oplus c_{i-1} \lor G_{i-1}$$ (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 $y = a_n \lor b_n \lor c_{n-1}$ and $\pi = a_n \land b_n \lor (a_n \lor b_n) \land c_{n-1}$.

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

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

$$r_n = \phi \left( (y, \pi) \odot (G_{n-1,1}, P_{n-1,1}) \right) \odot G_{n-1,1}$$ (7)

The latency of $\gamma$ and $\pi$ equations are justified as follows:

$$c_n \lor s_n \land c_{n-1} \lor G_{n-1,1}$$

$$= a_n \land b_n \lor (a_n \lor b_n) \land c_{n-1} \lor (a_n \lor b_n) \land \pi$$

$$= a_n \land b_n \lor (a_n \lor b_n) \land c_{n-1} \lor (a_n \lor b_n) \land c_{n-1} \land G_{n-1,1}$$

$$= G_{n-1,1} \lor c_{n-1} \lor G_{n-1,1}$$

$$= \phi \left( (y, \pi) \odot (G_{n-1,1}, P_{n-1,1}) \right) \odot G_{n-1,1}$$ (6)

REFERENCES

