two bit add algorithm

Consider the two bit add problem:

- Input:
  Two $n-$bit binary integers, stored in two $n-$element array A and B.
- Output:
  The sum of the two integers should be stored in binary form in an $(n+1)-$element array C.

You can find the implementation **here** or go to the next url: https://github.com/DiegoMendezMedina/C_Algorithms/tree/master/Bit_add/two_bit_add/implementations.

Pseudocode

two_bit_add(A[ ], B[ ])

1. $aux[1] = 0$
2. **for** $i = 1$ **to** n
3. $\quad C[i] = binary\_add(A[i], B[i], aux[1])$
4. $C[n+1] = aux$

binary_add(a, b, aux[1])

1. $sum = a + b + aux[1]$
2. **if** $sum >= 2$
3. $\quad aux[1] = 1$
4. **else** $aux[1] = 0$
5. **if** $sum == 2$
6. $\quad$ **return** 0
7. **return** 1

Proof

binary_add
This algorithm returns the binary number result of adding the two binary numbers received and saves the carrying on the aux array.
There are four different cases fora two binary add:

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} 0 \\ +\ 0 \\ \hline 0\ 0 \end{array}$$

Now lets see the binary add with carrying(purple):

$$\begin{array}{r} \mathbf{0} \\ 1 \\ +\ 1 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} \mathbf{0} \\ 1 \\ +\ 0 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} \mathbf{0} \\ 0 \\ +\ 1 \\ \hline 0\ 1 \end{array} \qquad \begin{array}{r} \mathbf{0} \\ 0 \\ +\ 0 \\ \hline 0\ 0 \end{array}$$

$$\begin{array}{r} \mathbf{1} \\ +\ 1 \\ \hline 1\ \mathbf{0} \\ +\ 1 \\ \hline 1\ 1 \end{array} \qquad \begin{array}{r} \mathbf{1} \\ +\ 1 \\ \hline 1\ \mathbf{0} \\ +\ 0 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} \mathbf{1} \\ +\ 0 \\ \hline 0\ \mathbf{1} \\ +\ 1 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} \mathbf{1} \\ +\ 0 \\ \hline 0\ \mathbf{1} \\ +\ 0 \\ \hline 0\ 1 \end{array}$$

Now it's clear to see :

- The carrying is 1 $\iff$ the *natural* add between three digits is ≥ 2. (lines 2-3)
- And zero otherwhise. (line 4)
- Red numbers are returned only when the *natural* add between three digits is zero. (line 5-6)
- One is returned in other cases. (line 7)

two_bit_add
Loop invariant:
For each iteration of the **for** loop on the C array it's stored the binary add of the two digits with the same index of the array A and B (lines 2-3)
Initialization:
Since there haven't been any add yet, the aux is set on zero. (line 1)
On the first iteration on $C[1]$ is stored the binary add of $A[1]$, $B[1]$, and zero. The corresponding carrying is set on aux[1].
Maintenance:
On the rest of the iteration for every on C[i] is stored the binary add of $A[i]$, $B[i]$ and the carrying calculated on the previous binary add. The corresponding carrying is set on aux[1].
Termination:
When the loop finishes **i** had browsed A and B and stored the corresponding digits on C. There's still one digit that hasn't been stored in C, the carrying of the $n-1$ add. and that's done on the line 4.