```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 20 des épreuves pratiques NSI 2022
"""


from doctest import testmod


def xor(tab1: list[int], tab2: list[int]) -> list[int]:
    """
    La fonction xor qui prend en paramètres deux tableaux
    de même longueur et qui renvoie un tableau où l¿élément
    situé à position i est le résultat, par l¿opérateur
    « ou exclusif », des éléments à la position i des
    tableaux passés en paramètres

    Args:
        tab1 (list[int]): tableau de 0 et de 1
        tab2 (list[int]): tableau de 0 et de 1 (même taille que tab1)

    Returns:
        list[int]: XOR entre chaque élément i de tab1 et tab2

    Tests et exemples:
    >>> a = [1, 0, 1, 0, 1, 1, 0, 1]
    >>> b = [0, 1, 1, 1, 0, 1, 0, 0]
    >>> xor(a, b)
    [1, 1, 0, 1, 1, 0, 0, 1]
    >>> c = [1, 1, 0, 1]
    >>> d = [0, 0, 1, 1]
    >>> xor(c, d)
    [1, 1, 1, 0]
    """
    nb_bits = len(tab1)

    # création du tableau résultat
    # (pour l'instant plein de vide)
    tab_xor = [None] * nb_bits

    # parcours des tableaux tab1 et tab2 bit par bit
    for i in range(nb_bits):
        bit1 = tab1[i]
        bit2 = tab2[i]

        # pour éviter de faire 4 tests, on remarque que
        # XOR vaut 0 <=> bit1 et bit2 sont différents !
        if bit1 == bit2:
            tab_xor[i] = 0
        else:
            tab_xor[i] = 1

    return tab_xor


a = [1, 0, 1, 0, 1, 1, 0, 1]
b = [0, 1, 1, 1, 0, 1, 0, 0]
c = [1, 1, 0, 1]
d = [0, 0, 1, 1]


# tests avec des affichages
print(xor(a, b))
print(xor(c, d))


# tests avec des assertions
assert(xor(a, b) == [1, 1, 0, 1, 1, 0, 0, 1])
assert(xor(c, d) == [1, 1, 1, 0])


# tests de la fonction avec doctest
testmod()
```