Voici la solution pour modéliser un **additionneur 4 bits** basé sur l'additionneur 1 bit, ainsi que le test via un banc de test.

2.1 Organisation

L'additionneur 4 bits sera composé de 4 additionneurs 1 bit connectés en série.
Chaque additionneur utilise la sortie de retenue (**COUT**) du précédent comme entrée de retenue (**CIN**).

Schéma fonctionnel (texte):

- Les signaux `A[3:0]` et `B[3:0]` sont les entrées de 4 bits.
- Le signal `CY_IN` est la retenue d'entrée (CIN_4).
- Les sorties sont `S[3:0]` (somme 4 bits) et `CY_OUT` (retenue de sortie).

2.2 Description

Entité VHDL :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ADD4B is
    generic (
        TP: time := 10 ns -- Temps de propagation par défaut
    );
    Port (
        A4     : in  STD_LOGIC_VECTOR(3 downto 0);
        B4     : in  STD_LOGIC_VECTOR(3 downto 0);
        CIN_4  : in  STD_LOGIC;
        S4     : out STD_LOGIC_VECTOR(3 downto 0);
        COUT_4 : out STD_LOGIC
    );
end ADD4B;
```

Architecture VHDL :

```
architecture Behavioral of ADD4B is
    signal COUT: STD_LOGIC_VECTOR(3 downto 0); -- Retenues internes
begin
```

```
    -- Instanciation des 4 additionneurs 1 bit
    U0: entity work.ADD1B
        generic map (TP => TP)
        port map (A => A4(0), B => B4(0), CIN => CIN_4, S => S4(0), COUT => COUT(0));

    U1: entity work.ADD1B
        generic map (TP => TP)
        port map (A => A4(1), B => B4(1), CIN => COUT(0), S => S4(1), COUT => COUT(1));

    U2: entity work.ADD1B
        generic map (TP => TP)
        port map (A => A4(2), B => B4(2), CIN => COUT(1), S => S4(2), COUT => COUT(2));

    U3: entity work.ADD1B
        generic map (TP => TP)
        port map (A => A4(3), B => B4(3), CIN => COUT(2), S => S4(3), COUT => COUT_4);
end Behavioral;
```

2.3 Test (Testbench)

Le banc de test décrit dans l'énoncé peut être amélioré pour tester exhaustivement le comportement de l'additionneur.

Banc de test VHDL :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB_ADD4B is
end TB_ADD4B;

architecture Test of TB_ADD4B is
    signal A, B, S: STD_LOGIC_VECTOR(3 downto 0);
    signal CIN, COUT: STD_LOGIC;
begin
    -- Instanciation de l'additionneur 4 bits
    UUT: entity work.ADD4B
        generic map (TP => 10 ns) -- Temps de propagation paramétré
        port map (
            A4 => A,
            B4 => B,
            CIN_4 => CIN,
            S4 => S,
```

```
        COUT_4 => COUT
    );

  -- Stimuli de test
  process
  begin
    -- Tester plusieurs combinaisons
    A <= "0000"; B <= "0000"; CIN <= '0'; wait for 100 ns;
    A <= "0010"; B <= "0011"; CIN <= '0'; wait for 100 ns;
    A <= "0100"; B <= "0101"; CIN <= '1'; wait for 100 ns;
    A <= "0110"; B <= "0111"; CIN <= '0'; wait for 100 ns;
    A <= "1000"; B <= "1001"; CIN <= '1'; wait for 100 ns;
    A <= "1010"; B <= "1011"; CIN <= '0'; wait for 100 ns;
    A <= "1100"; B <= "1101"; CIN <= '1'; wait for 100 ns;
    A <= "1110"; B <= "1111"; CIN <= '0'; wait for 100 ns;
    wait;
  end process;
end Test;
```

Simulation et résultats

1. **Lancer la simulation :**
    - Compilez le code dans un outil comme **ModelSim** ou **Vivado**.
    - Observez les signaux de sortie `S[3:0]` et `COUT`.
2. **Problèmes potentiels liés au temps de propagation :**
    - Si le temps de propagation (TP) est significatif, la valeur correcte de `COUT` risque d'être légèrement décalée.
    - Cela peut entraîner des instabilités temporaires sur les bits de poids fort dans `S[3:0]` jusqu'à la stabilisation complète.

Souhaitez-vous approfondir la simulation ou passer à l'additionneur 16 bits ? 😊