Voici une solution structurée pour votre travail pratique (TP) sur la modélisation d'un additionneur combinatoire en VHDL.

---

1. ADDITIONNEUR 1 BIT

1.1 Description en VHDL

Entité:

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

entity ADD1B is
    generic (
        TP: time := 10 ns -- Temps de propagation par défaut
    );
    Port (
        A     : in  STD_LOGIC;
        B     : in  STD_LOGIC;
        CIN   : in  STD_LOGIC;
        S     : out STD_LOGIC;
        COUT  : out STD_LOGIC
    );
end ADD1B;
```

Architecture:

```
architecture Behavioral of ADD1B is
begin
    process (A, B, CIN)
    begin
        S <= (A xor B xor CIN) after TP;
        COUT <= (A and B) or (B and CIN) or (A and CIN) after TP;
    end process;
end Behavioral;
```

1.2 Simulation

Pour tester le fonctionnement, créez un banc de test (testbench).

Testbench VHDL:

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

entity TB_ADD1B is
end TB_ADD1B;

architecture Test of TB_ADD1B is
   -- Signaux de test
   signal A, B, CIN, S, COUT: STD_LOGIC;
begin
   -- Instanciation de l'additionneur 1 bit
   UUT: entity work.ADD1B
      generic map (
         TP => 10 ns -- Temps de propagation
      )
      port map (
         A => A,
         B => B,
         CIN => CIN,
         S => S,
         COUT => COUT
      );

   -- Stimuli de test
   process
   begin
      -- Tester tous les cas possibles (2^3 = 8 combinaisons)
      A <= '0'; B <= '0'; CIN <= '0'; wait for 20 ns;
      A <= '0'; B <= '1'; CIN <= '0'; wait for 20 ns;
      A <= '1'; B <= '0'; CIN <= '0'; wait for 20 ns;
      A <= '1'; B <= '1'; CIN <= '0'; wait for 20 ns;
      A <= '0'; B <= '0'; CIN <= '1'; wait for 20 ns;
      A <= '0'; B <= '1'; CIN <= '1'; wait for 20 ns;
      A <= '1'; B <= '0'; CIN <= '1'; wait for 20 ns;
      A <= '1'; B <= '1'; CIN <= '1'; wait for 20 ns;
      wait;
   end process;
end Test;
```

Exécution des tests:

- Simulez le testbench avec un outil comme **ModelSim**, **Vivado** ou tout autre simulateur VHDL.
- Observez la sortie pour vérifier que la table de vérité est respectée.

- Notez comment le temps de propagation (TP) affecte les transitions sur les sorties.

Résumé des résultats attendus:

1. Le **signal S** doit correspondre à `A XOR B XOR CIN`.
2. Le **signal COUT** doit correspondre à `A AND B OR B AND CIN OR A AND CIN`.
3. Les transitions doivent être décalées en fonction de la valeur de **TP**.

Souhaitez-vous que l'on continue avec l'additionneur 16 bits en utilisant cette base ? 😊