

TP 2 : Modélisation d'un circuit combinatoire simple

Objectifs: - créer une structure à partir d'éléments simples

- utiliser pour cela l'instanciation de composants

- utiliser la notion de généricité (pour les temps de propagation)

- réaliser une unité de test en VHDL

Le circuit envisagé est un additionneur 16 bits avec retenue à l'entrée. On se propose de le réaliser à partir d'additionneurs élémentaires qui seront compilés et testés séparément.

La structure du circuit final est donnée Figure 1

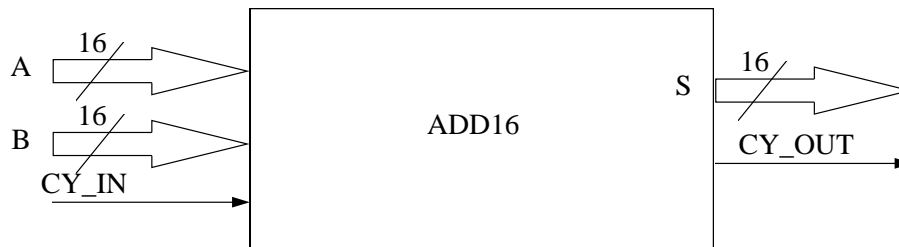


Figure 1 Additionneur 16 bits

1. ADDITIONNEUR 1 BIT:

Le schéma de base pour ce type de composant est donné sur la Figure 2

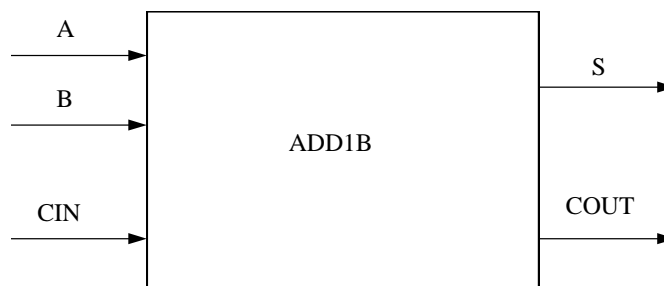


Figure 2 Additionneur 1 bit

La table de vérité d'un tel système est la suivante:

A	B	CIN	S	COUT
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Les sorties S et COUT sont donnés par :

$$S = A \oplus B \oplus CIN$$

$$COUT = A.B + B.CIN + A.CIN$$

1.1. Description en VHDL

Ecrire l'entité et l'architecture correspondante en prévoyant un temps de propagation générique, qui pourra être modifié lors d'une instanciation ultérieure.

1.2. Simulation

Vérifier le fonctionnement du circuit à l'aide du simulateur, en testant tous les cas possibles, et en notant l'influence du temps propagation.

2. Additionneur 4 bits

2.1. Organisation

Proposer un schéma permettant de réaliser un additionneur 4 bits à partir de composant précédent; il s'agit finalement d'établir les liaisons internes du bloc suivant.

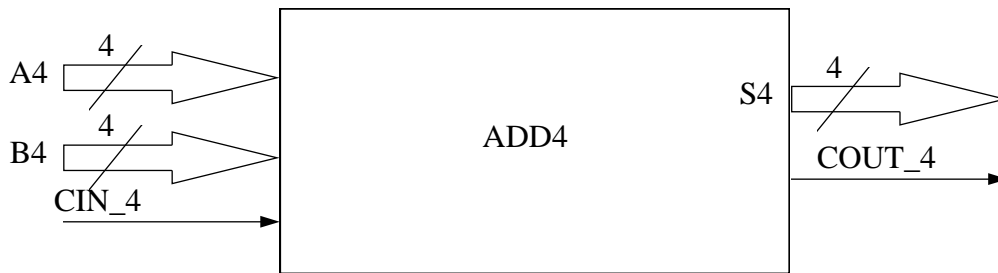


Figure 3 Additionneur 4 bits

2.2. Description

Proposer un couple entité/.architecture pour décrire l'additionneur; on veillera à permettre le paramétrage du temps de propagation au moment de l'instanciation.

2.3. Test

La technique préconisée est celle du fichier de simulation VHDL encore appelé *testbench*. C'est une architecture qui instancie le circuit testé, fournit les stimuli et vérifie les résultats.

```

-- exemple de test de ADD4B
entity TEST is
end TEST;

architecture A_TEST of TEST is

    signal A,B,S      : bit_vector(0 to 3);      -- déclarations des signaux utiles
    signal CY_IN,CY_OUT : bit;

    component ADD4B                                     -- déclaration du composant
    port(A4,B4 : in bit_vector(0 to 3);
          CIN_4 : in bit;
          S4   : out bit_vector(0 to 3);
          COUT_4: out bit);
    end component;

    begin
        instance: ADD4B                                -- instanciation du circuit
        port map(A,B,CY_IN,S,CY_OUT);

        -- génération des stimuli
        A<="0000" after 0 ns,"0010" after 200 ns,"0100" after 400 ns,"0110" after 600
ns,"1000" after 800 ns,
  
```

```
"1010" after 1000 ns,"1100" after 1200 ns,"1110" after 1400 ns;  
  
B<="0000" after 0 ns,"0011" after 300 ns,"0101" after 500 ns,"0111" after 700 ns,  
"1001" after 900 ns,"1011" after 1100 ns,"1101" after 1300 ns,"1111" after 1500 ns;  
  
CY_IN<='0' after 0 ns, '1' after 750 ns;  
  
end A_TEST;
```

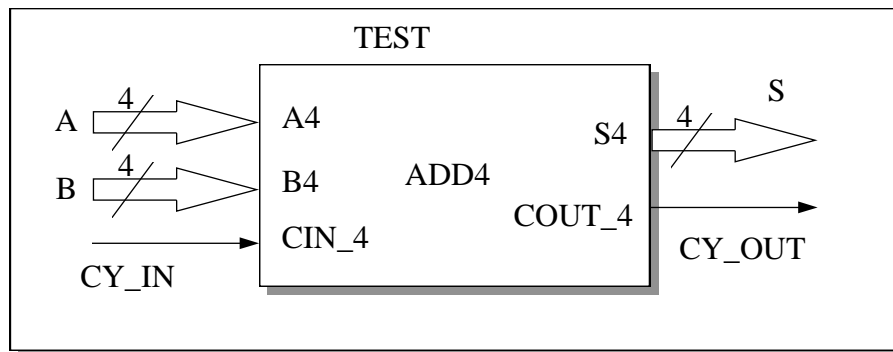


Figure 4 Architecture de test

Après compilation, il suffit de lancer la simulation pour le *composant* test.

Faire apparaître dans les résultats de simulation les problèmes rencontrés relatifs au temps de propagation de la retenue pendant le calcul.

3. Additionneur 16 bits

En utilisant les techniques décrites précédemment, décrire puis tester un additionneur 16 bits. Fournir les textes VHDL, ainsi que les résultats des simulations obtenues.