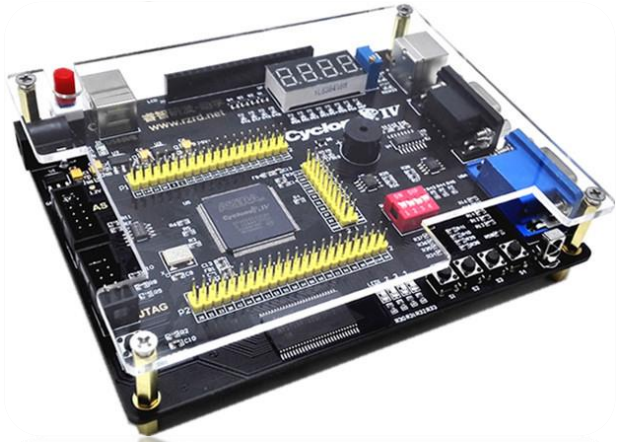
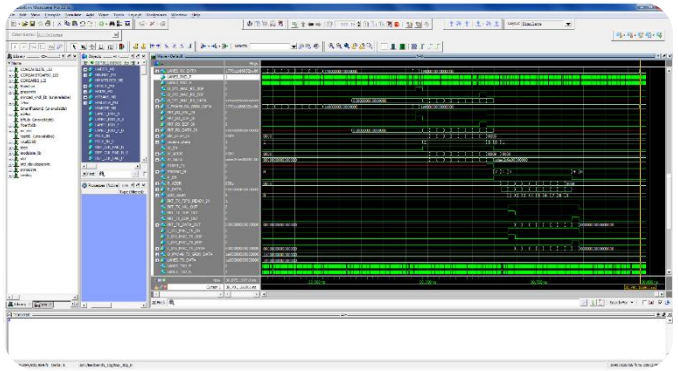
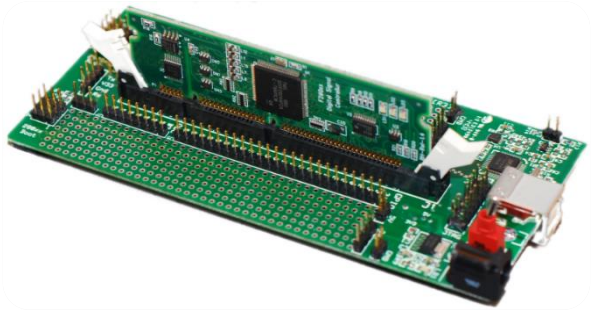
Cours FPGA, VHDL et DSP

Master d'Université Spécialisé en Ingénierie des Systèmes Embarqués (ISE)





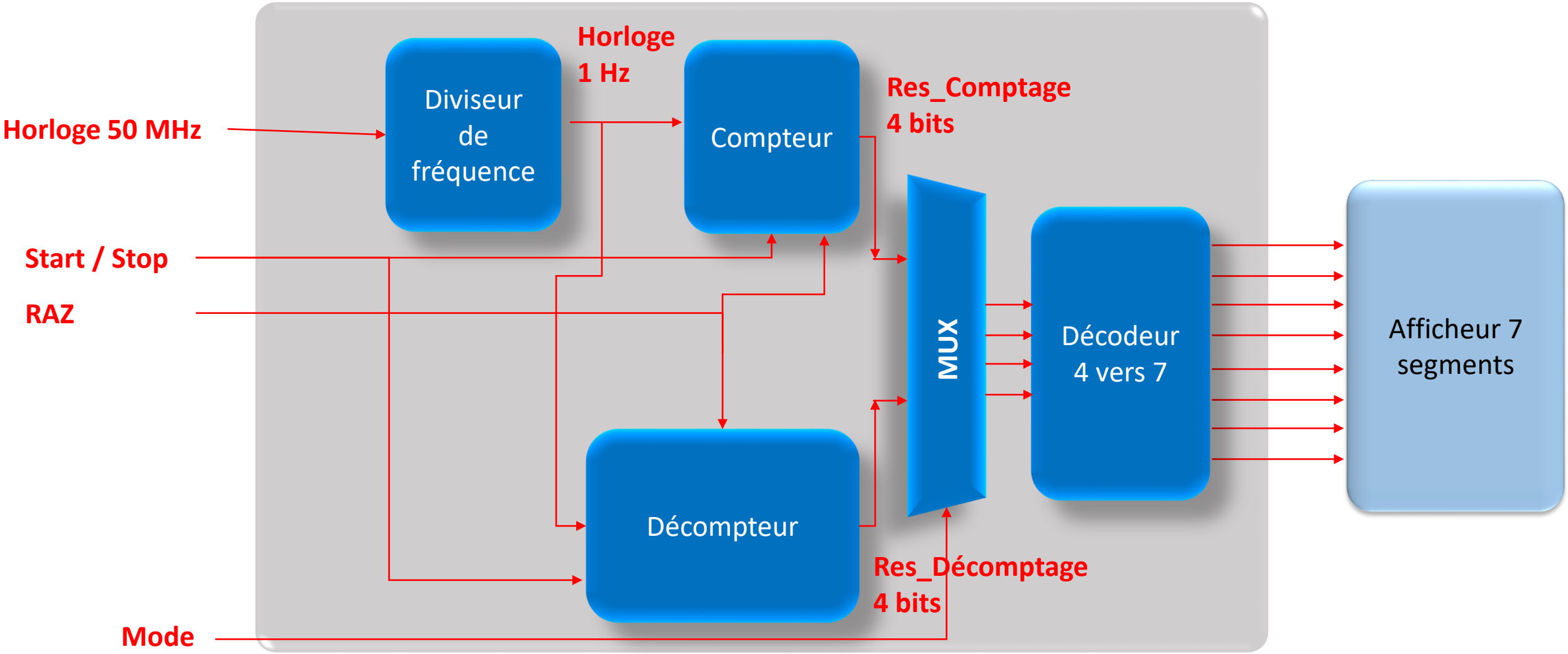


Année universitaire 2024/2025

Dr. Ing. LASSIOUI Abdellah

Chapitre 5

Objectifs du chapitre N5: Implémenter un système numérique selon la méthodologie de conception structurelle TOP – DOWN



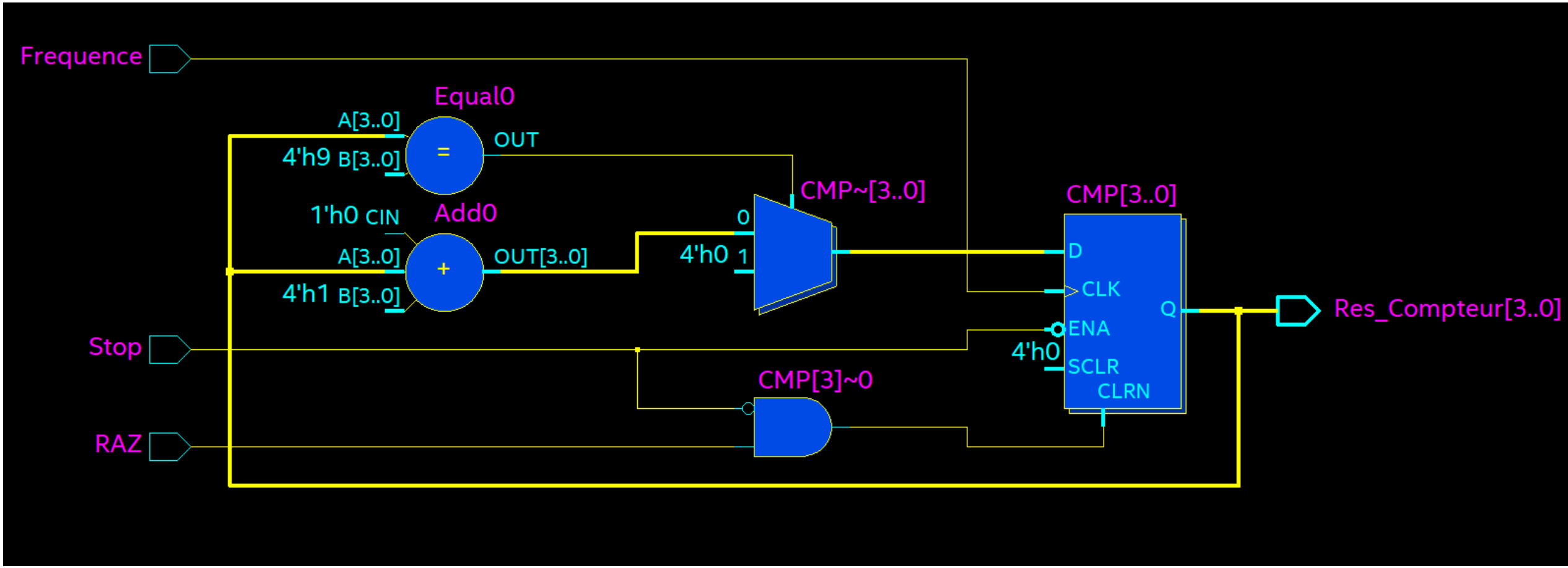
Code décrivant le fonctionnement du compteur

- **Ce code permet de décrire le fonctionnement du compteur**

Compteur.vhd | Compilation Report - Compteur | Compteur_TB.v

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity Compteur is
5  port(
6   Stop : in std_logic;
7   RAZ : in std_logic;
8   Frequence : in std_logic;
9  Res_Compteur: out std_logic_vector (3 downto 0));
10 end entity;
11 architecture arc of Compteur is
12  signal CMP : std_logic_vector (3 downto 0):="0000";
13 begin
14 process (Stop, RAZ, Frequence) is
15  begin
16 if (Stop = '1') then
17  CMP <= CMP;
18 elsif (RAZ ='1') then
19  CMP <= "0000";
20 elsif (Frequence'event and Frequence ='1') then
21 if (CMP = "1001") then
22  CMP <= "0000";
23 else
24  CMP <= unsigned (CMP) + 1;
25 end if;
26 end if;
27 Res_Compteur <= CMP;
28 end process;
29 end arc;
30
```

Schéma RTL du compteur généré par l'outil Netlist Viewer



Code de test du fonctionnement du compteur (TestBench)

- **Ce deuxième code permet de tester le fonctionnement compteur réalisée précédemment**

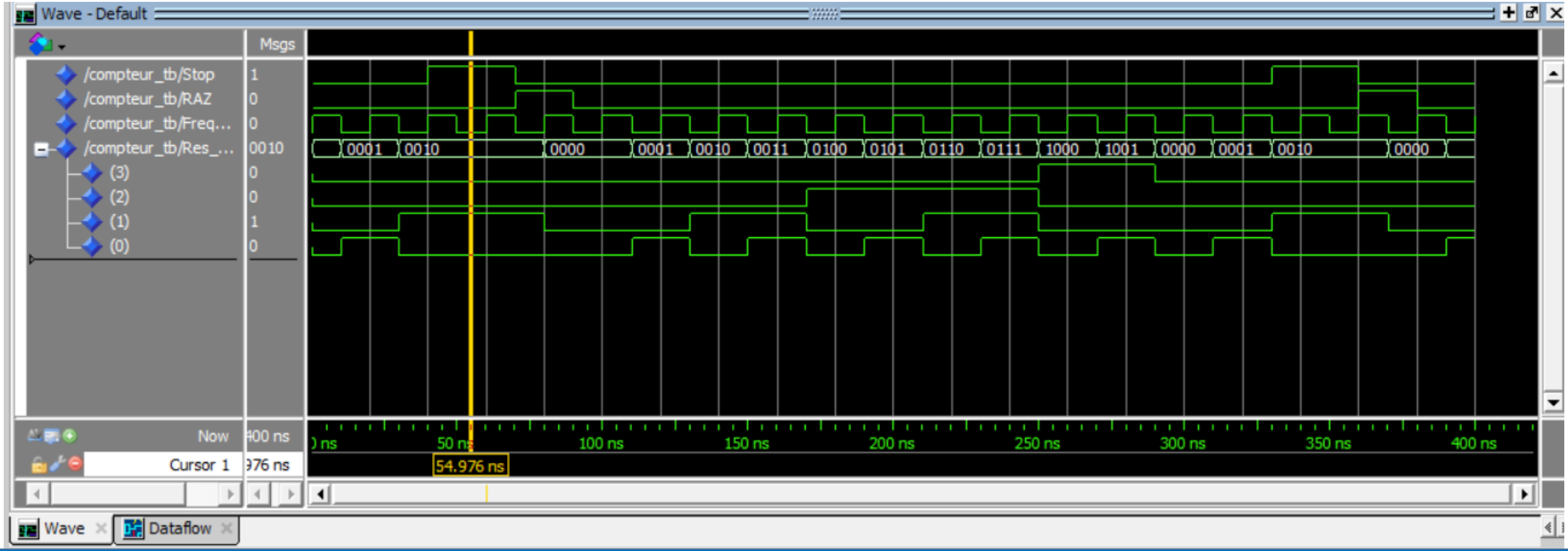
Compteur.vhd | Compilation Report - Compteur | Compteur_TB.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity Compteur_TB is
5  end entity;
6  architecture arc of Compteur_TB is
7  signal Stop : std_logic:='0';
8  signal RAZ : std_logic:='0';
9  signal Frequence :  std_logic :='0';
10 signal Res_Compteur: std_logic_vector (3 downto 0):="0000";
11
12 component Compteur is
13 port(
14 Stop : in std_logic;
15 RAZ : in std_logic;
16 Frequence : in std_logic;
17 Res_Compteur: out std_logic_vector (3 downto 0));
18 end component;
19 begin
20 instantiation : Compteur
21 port map (Stop => Stop, RAZ => RAZ, Frequence => Frequence, Res_Compteur => Res_Compteur);
22 clk: process is
23 begin
```

```
24 Frequence <= not Frequence;
25 wait for 10 ns;
26 end process;
27 test : process is
28 begin
29 Stop <= '0';
30 RAZ <= '0';
31 wait for 40 ns;
32 Stop <= '1';
33 RAZ <= '0';
34 wait for 30 ns;
35 Stop <= '0';
36 RAZ <= '1';
37 wait for 20 ns;
38 Stop <= '0';
39 RAZ <= '0';
40 wait for 200 ns;
41 end process;
42 end arc;
```

Résultas de la simulation sous ModelSim

- **Le fonctionnement du compteur est bien vérifié comme le montre la figure suivante:**
  - **Si le signal Stop est à 1 alors le comptage est arrêté**
  - **Si le signal RAZ est à 1 alors le comptage est réinitialisé**
  - **Sinon le comptage continue jusqu'au 1001 puis il est remis à 0**



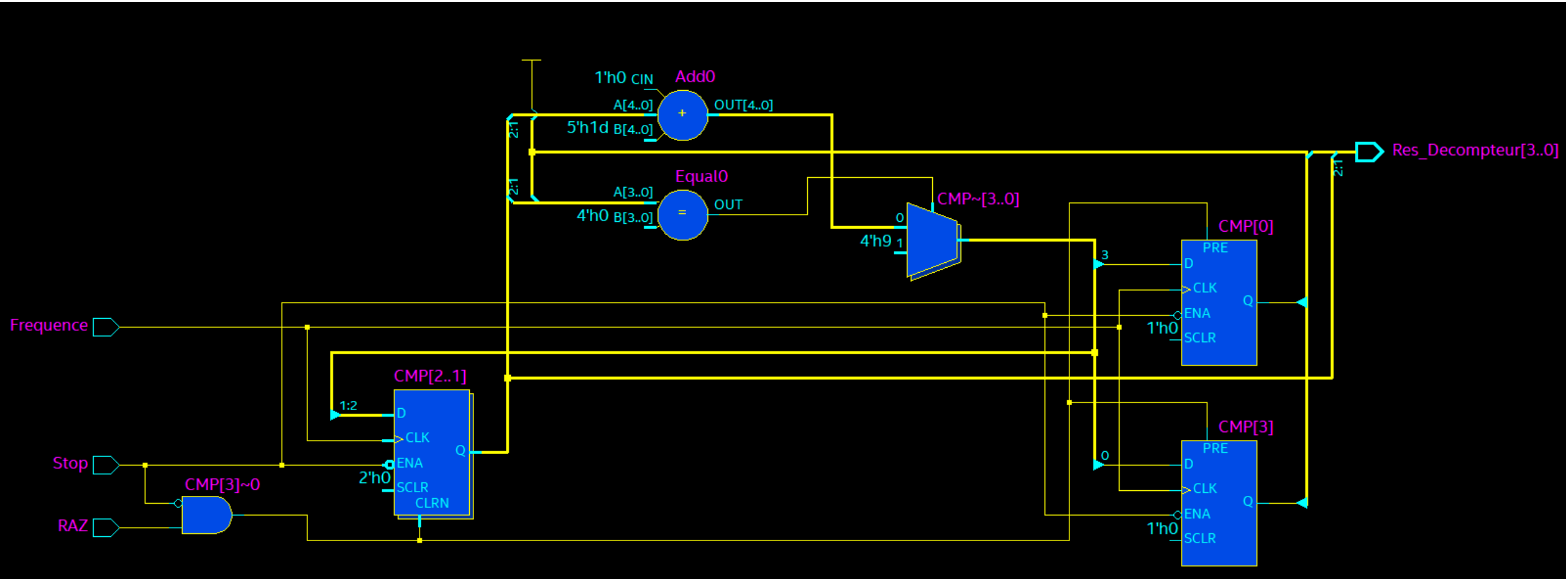
Code décrivant le fonctionnement du décompteur

- **Ce code permet de décrire le fonctionnement du décompteur**

Compteur.vhd | Compteur_TB.vhd | Decompteur.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity Decompteur is
5  port(
6  Stop : in std_logic;
7  RAZ : in std_logic;
8  Frequence : in std_logic;
9  Res_Decompteur: out std_logic_vector (3 downto 0));
10 end entity;
11 architecture arc of Decompteur is
12 signal CMP : std_logic_vector (3 downto 0):="0000";
13 begin
14 process (Stop, RAZ, Frequence) is
15 begin
16 if (Stop = '1') then
17 CMP <= CMP;
18 elsif (RAZ ='1') then
19 CMP <= "1001";
20 elsif (Frequence'event and Frequence ='1') then
21 if (CMP = "0000") then
22 CMP <= "1001";
23 else
24 CMP <= unsigned (CMP) - 1;
25 end if;
26 end if;
27 Res_Decompteur <= CMP;
28 end process;
29 end arc;
30
```

Schéma RTL du décompteur généré par l'outil Netlist Viewer



Code de test du fonctionnement du décompteur (TestBench)

- **Ce deuxième code permet de tester le fonctionnement du décompteur réalisée précédemment**

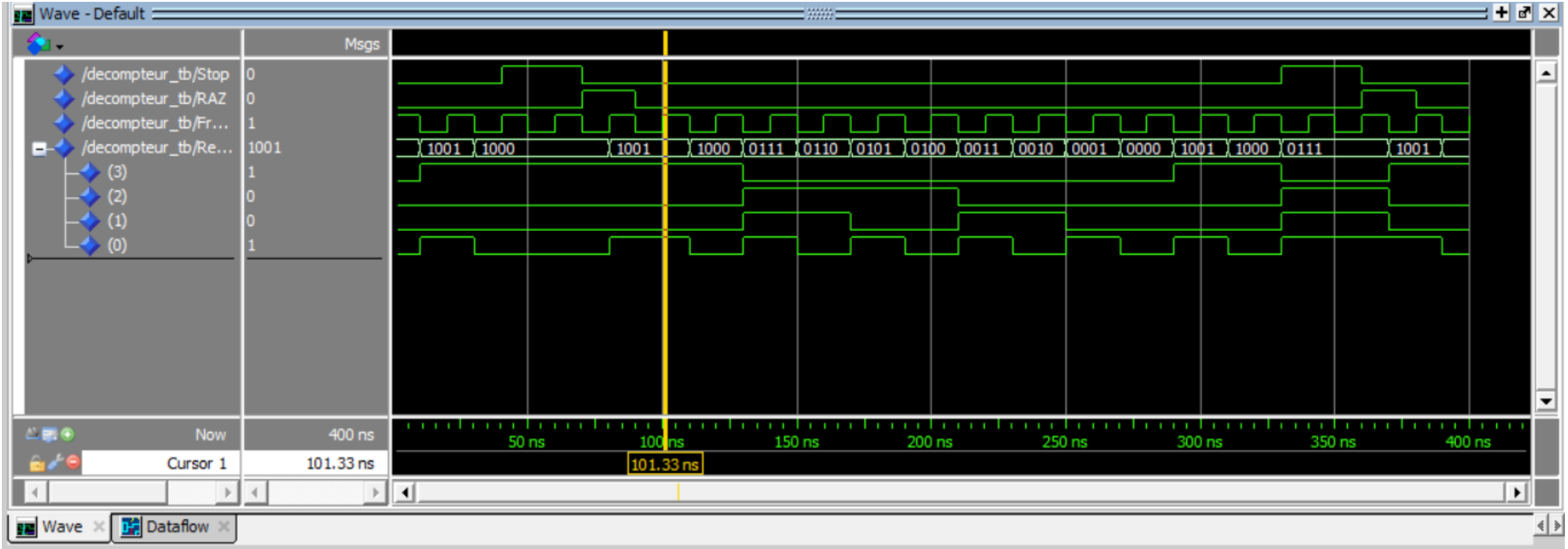
Compteur.vhd | Compteur_TB.vhd | Decompteur.vhd | Decompteur_TB.vhd | Compilation Report - Compteur

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity Decompteur_TB is
5  end entity;
6  architecture arc of Decompteur_TB is
7   signal Stop : std_logic:='0';
8   signal RAZ : std_logic:='0';
9   signal Frequence :  std_logic :='0';
10  signal Res_Decompteur: std_logic_vector (3 downto 0):="0000";
11
12  component Decompteur is
13  port(
14  Stop : in std_logic;
15  RAZ : in std_logic;
16  Frequence : in std_logic;
17  Res_Decompteur: out std_logic_vector (3 downto 0));
18  end component;
19  begin
20  instantiation : Decompteur
21  port map (Stop => Stop, RAZ => RAZ, Frequence => Frequence, Res_Decompteur => Res_Decompteur);
22  clk: process is
23  begin
```

```
24  Frequence <= not Frequence;
25  wait for 10 ns;
26  end process;
27  test : process is
28  begin
29  Stop <= '0';
30  RAZ <= '0';
31  wait for 40 ns;
32  Stop <= '1';
33  RAZ <= '0';
34  wait for 30 ns;
35  Stop <= '0';
36  RAZ <= '1';
37  wait for 20 ns;
38  Stop <= '0';
39  RAZ <= '0';
40  wait for 200 ns;
41  end process;
42  end arc;
```

Résultas de la simulation sous ModelSim

- **Le fonctionnement du déompteur est bien vérifié comme le montre la figure suivante:**
  - **Si le signal Stop est à 1 alors le décomptage est arrêté**
  - **Si le signal RAZ est à 1 alors le décomptage est réinitialisé**
  - **Sinon le décomptage continu jusqu'au 0 puis il est remis à 1001**

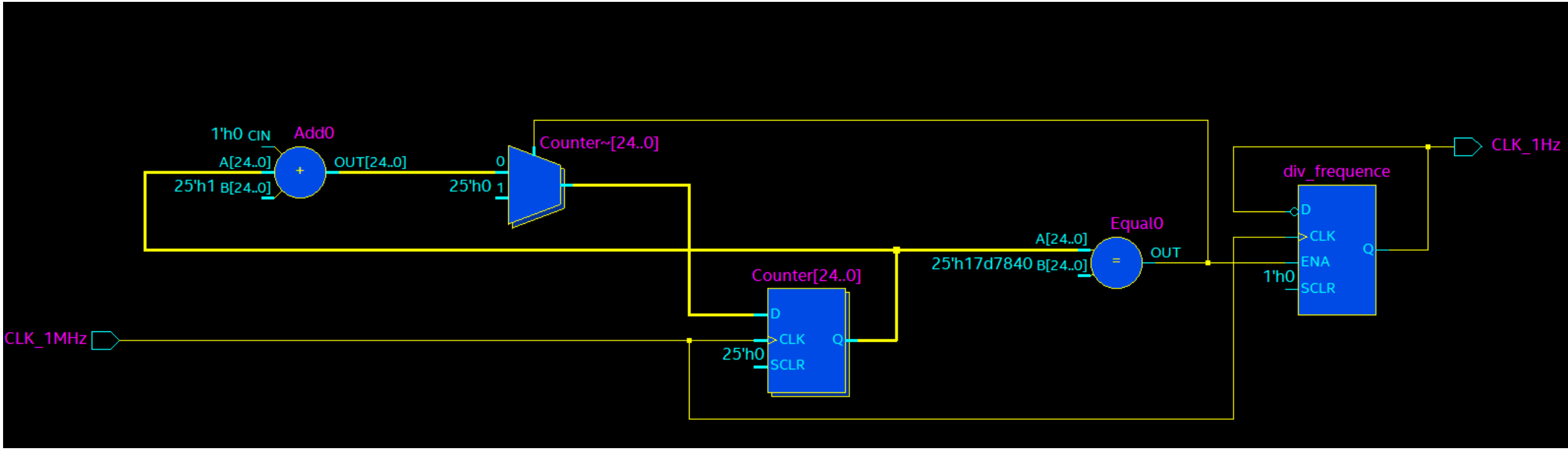


Code décrivant le fonctionnement du diviseur de fréquence

- **Ce code permet de décrire le fonctionnement du diviseur de fréquence**

```
3   use ieee.std_logic_arith.all;
4   entity Div_Frequence is
5   port (
6    CLK_1MHz : in std_logic;
7    CLK_1Hz : out std_logic);
8    end entity;
9   architecture arc of Div_Frequence is
10   SIGNAL div_frequence : std_logic :='0';
11  begin
12  process (CLK_1MHz)
13   variable Counter : integer range 0 to 25000000 :=0;
14   begin
15  if (CLK_1MHz'event and CLK_1MHz ='1') then
16  if (Counter = 25000000) then
17   Counter := 0;
18  div_frequence  <= not (div_frequence);
19  else
20   Counter := Counter + 1;
21  end if;
22  end if;
23   CLK_1Hz <= div_frequence;
24  end process;
25   end arc;
```

Schéma RTL du diviseur de fréquence généré par l'outil Netlist Viewer



Code de test du fonctionnement du décompteur (TestBench)

- **Ce deuxième code permet de tester le fonctionnement du diviseur de fréquence**

Decompteur.vhd | Decompteur_TB.vhd | Compilation Report - Compteur

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity Div_Frequence_TB is
5  end entity;
6  architecture arc of Div_Frequence_TB is
7  signal CLK_1MHz : std_logic := '0';
8  signal CLK_1Hz : std_logic := '0';
9  component Div_Frequence is
10 port (
11 CLK_1MHz : in std_logic;
12 CLK_1Hz : out std_logic);
13 end component;
14 begin
15 instantiation : Div_Frequence
16 port map (CLK_1MHz => CLK_1MHz, CLK_1Hz => CLK_1Hz);
17 process  is
18 begin
19 CLK_1MHz <= NOT (CLK_1MHz);
20 wait for 10 ns;
21 end process;
22 end arc;
```

Résultas de la simulation sous ModelSim

- **Le fonctionnement du diviseur de fréquence est bien vérifié comme le montre la figure suivante:**
- **Le signal CLK_1Hz change d'état chaque 0.5s**

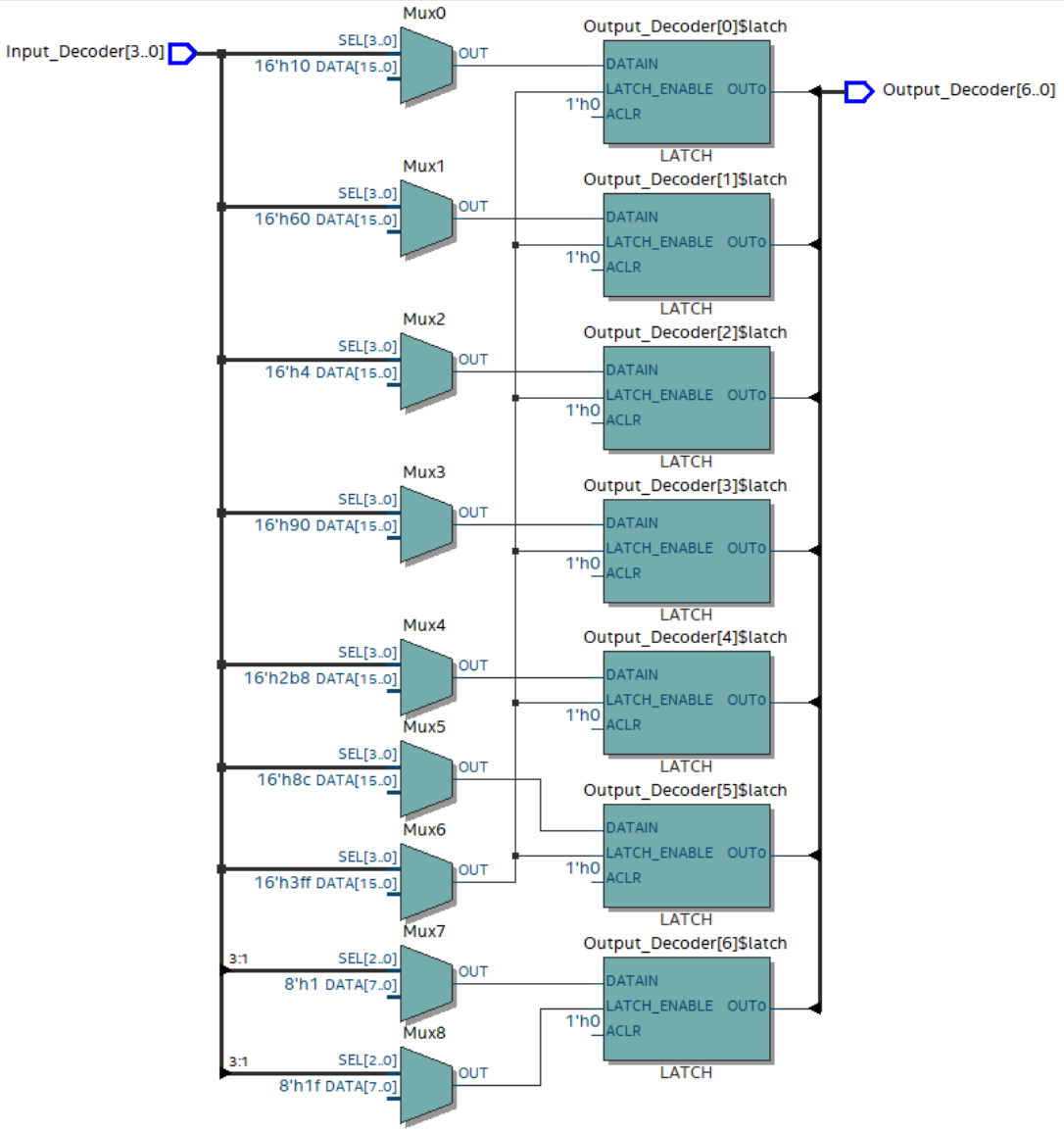
Code décrivant le fonctionnement du décodeur 7 Segments

- **Ce code permet de décrire le fonctionnement du décodeur 7 ségments**

ecompteur_TB.vhd | Compilation Report - Compteur | Div_Freque

```
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4   entity Decodeur_7seg is
5   port (
6   Input_Decoder: in std_logic_vector (3 downto 0);
7   Output_Decoder: out std_logic_vector (6 downto 0));
8   end entity;
9   architecture arc of Decodeur_7seg is
10  begin
11  process (Input_Decoder)
12   begin
13  case Input_Decoder is
14   when "0000" => Output_Decoder <= "1000000";
15   when "0001"=> Output_Decoder <= "1000000";
16   when "0010"=> Output_Decoder <= "0100100";
17   when "0011"=> Output_Decoder <= "0110000";
18   when "0100"=> Output_Decoder <= "0011001";
19   when "0101"=> Output_Decoder <= "0010010";
20   when "0110"=> Output_Decoder <= "0000010";
21   when "0111"=> Output_Decoder <= "0111000";
22   when "1000"=> Output_Decoder <= "0000000";
23   when "1001"=> Output_Decoder <= "0010000";
24   when others => NULL;
```

Schéma RTL du décodeur généré par l'outil Netlist Viewer



Code du système total

- **Ce deuxième code permet de décrire le fonctionnement du système total**

System_Total.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity System_Total is
5  port(
6  CLK_1MHZ : in std_logic;
7  RAZ_Total: in std_logic;
8  STOP_Total : in std_logic;
9  Mode : in std_logic;
10 Digit_Select: out std_logic;
11 Output_Total : out std_logic_vector (6 downto 0));
12 end entity;
13 architecture arc of System_Total is
14 signal CLK_1Hz: std_logic:='0';
15 signal Res_Compteur: std_logic_vector (3 downto 0) := "0000";
16 signal Res_Decompteur: std_logic_vector (3 downto 0) := "0000";
17 signal Input_Decoder: std_logic_vector (3 downto 0) := "0000";
18 --Signal Output_Decoder : std_logic_vector (6 downto 0) := "0000(
19 component Div_Frequence is
20 port (
21 CLK_1MHz : in std_logic;
22 CLK_1Hz : out std_logic);
23 end component;
24 component Compteur is
25 port(
26 Stop : in std_logic;
27 RAZ : in std_logic;
28 Frequence : in std_logic;
29 Res_Compteur: out std_logic_vector (3 downto 0));
30 end component;
31 component Decompteur is
32 port (
33 Stop : in std_logic;
34 RAZ : in std_logic;
35 Frequence : in std_logic;
36 Res_Decompteur: out std_logic_vector (3 downto 0));
37 end component;
38 Component Decodeur_7seg is
39 port (
40 Input_Decoder: in std_logic_vector (3 downto 0);
41 Output_Decoder: out std_logic_vector (6 downto 0));
42 end component;
43 begin
44 Module1 : Div_Frequence
45 port map (CLK_1MHz => CLK_1MHZ, CLK_1Hz => CLK_1Hz);
46 Module2 : Compteur
47 port map (Stop => STOP_Total, RAZ => RAZ_Total, Frequence => CLK_1Hz, Res_Compteur => Res_Compteur);
48 Module3 : Decompteur
49 port map (Stop => STOP_Total, RAZ => RAZ_Total, Frequence => CLK_1Hz, Res_Decompteur => Res_Decompteur);
50 Module4 : Decodeur_7seg
51 port map (Input_Decoder => Input_Decoder, Output_Decoder => Output_Total);
52 process (CLK_1Hz, RAZ_Total, Stop_Total, Res_Compteur, Res_Decompteur, Mode)
53 begin
54 case Mode is
55 when '0' => Input_Decoder <= Res_Compteur;
56 when '1' => Input_Decoder <= Res_Decompteur;
57 when others => Input_Decoder <= "ZZZZ";
58 end case;
59 end process;
60 Digit_Select <= '0';
61 end architecture;
```

Le schema RTL du sytsème total