Mohamed Manessouri
SSE

Tp5 : Modélisation Séquentielle

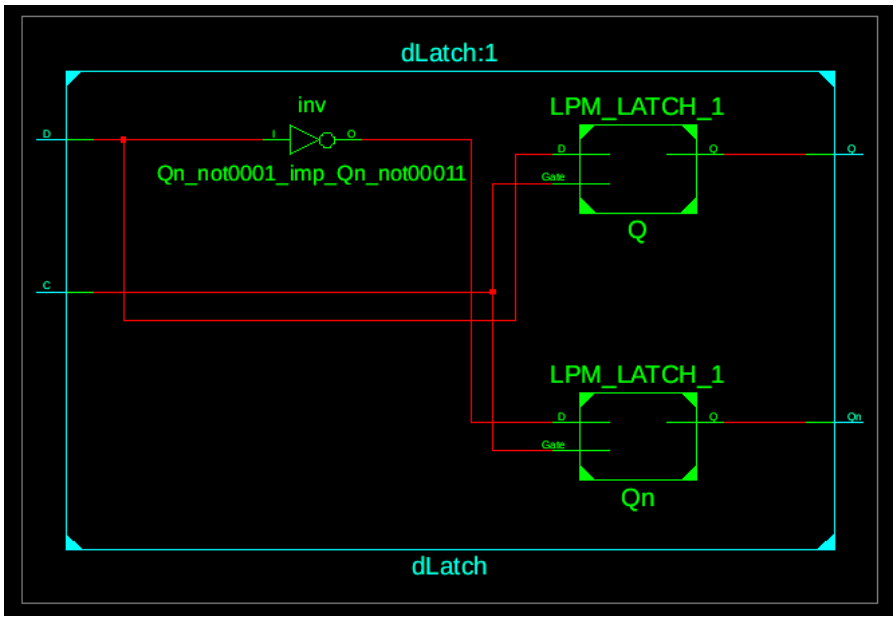
Stockage et registre

Exercice 1 :

1) Modèle VHDL modélisant les types de bascule D et test bench :

D_Latch

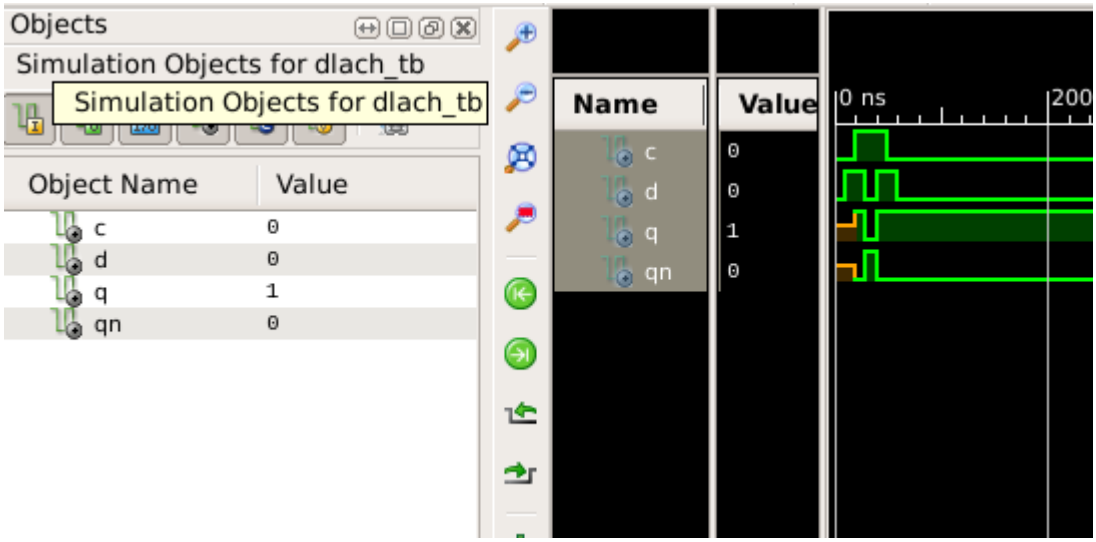
```
32  entity dLatch is
33      Port ( C : in  STD_LOGIC;
34             D : in  STD_LOGIC;
35             Q : out  STD_LOGIC;
36             Qn : out  STD_LOGIC);
37  end dLatch;
38
39  architecture Behavioral of dLatch is
40
41  begin
42  process(D,C)
43  begin
44    if C = '1' then
45       Q <= D;
46       QN <= not D;
47    end if;
48  end process;
49
50  end Behavioral;
51
52
```



Test bench pour D_lach:

```
34
35  ENTITY Dlach_tb IS
36  END Dlach_tb;
37
38  ARCHITECTURE behavior OF Dlach_tb IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT dLatch
43      PORT(
44           C : IN  std_logic;
45           D : IN  std_logic;
46           Q : OUT  std_logic;
47           Qn : OUT  std_logic
48          );
49      END COMPONENT;
50
51
52     --Inputs
53     signal C : std_logic := '0';
54     signal D : std_logic := '0';
55
56     --Outputs
57     signal Q : std_logic;
58     signal Qn : std_logic;
59
60
61
62  BEGIN
63
64     -- Instantiate the Unit Under Test (UUT)
65     uut: dLatch PORT MAP (
66           C => C,
67           D => D,
68           Q => Q,
69           Qn => Qn
```

```
68            Q => Q,
69            Qn => Qn
70          );
71
72     process
73     begin
74     D<= '0'; C <= '0'; wait for 10 ns;
75     D <= '1'; wait for 10 ns;
76
77     C <= '1'; wait for 10 ns;
78     D <= '0'; wait for 10 ns;
79     D <= '1'; wait for 10 ns;
80
81     C <= '0'; wait for 10 ns;
82     D <= '0'; wait for 10 ns;
83
84     wait;
85     end process;
86
87  END;
88
```

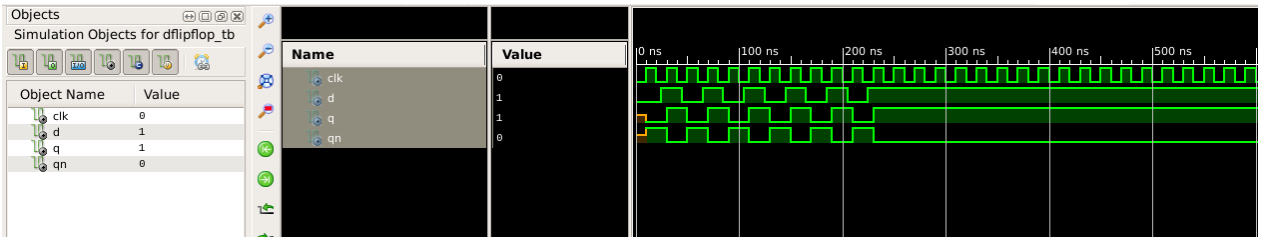


D_Flip_Flop

```
31
32  entity dFlipFlop is
33      Port ( Clk : in  STD_LOGIC;
34             D : in  STD_LOGIC;
35             Q : out  STD_LOGIC;
36             Qn : out  STD_LOGIC);
37  end dFlipFlop;
38
39  architecture Behavioral of dFlipFlop is
40
41  begin
42   process (Clk)
43      begin
44          if rising_edge(Clk) then
45              Q <= D;
46              Qn <= not D;
47          end if;
48      end process;
49
50  end Behavioral;
51
```

Testbench pour D_Flip_Flop:

```
71      Clk_Process: process
72       begin
73            Clk <= '0';
74            wait for 10 ns;
75            Clk <= '1';
76            wait for 10 ns;
77       end process;
78
79       Stimulus : process
80       begin
81       D <= '0'; wait for 25 ns;
82       D <= '1'; wait for 20 ns;
83       D <= '0'; wait for 20 ns;
84       D <= '1'; wait for 20 ns;
85       D <= '0'; wait for 20 ns;
86       D <= '1'; wait for 20 ns;
87       D <= '0'; wait for 20 ns;
88       D <= '1'; wait for 20 ns;
89       D <= '0'; wait for 20 ns;
90       D <= '1'; wait for 20 ns;
91       D <= '0'; wait for 20 ns;
92       D <= '1'; wait for 20 ns;
93       wait;
94
95       end process;
96
97  END;
```



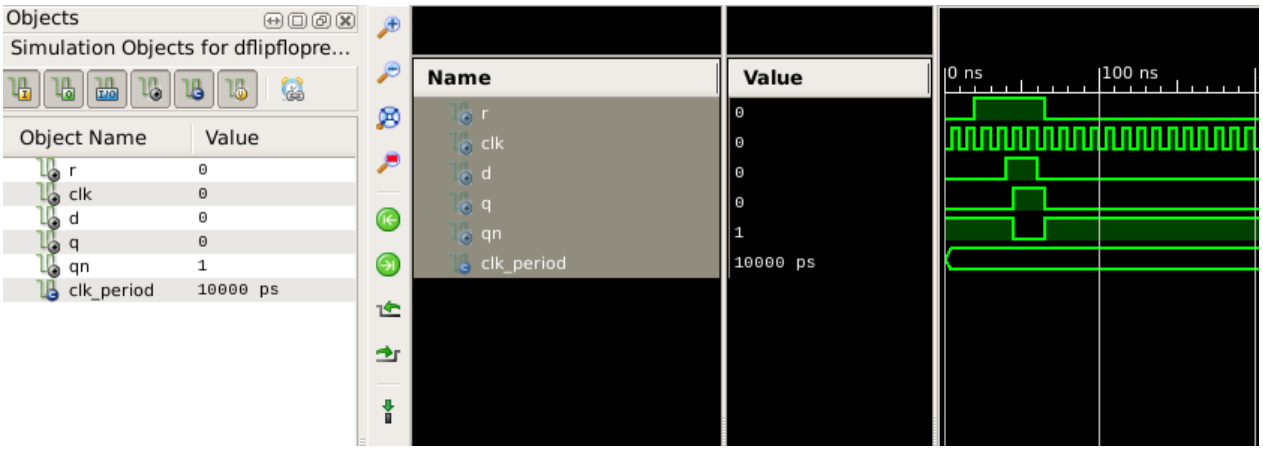
D_Flip_Flop avec réinitialisations asynchrones

```
entity DflipFLopReinAssync is
    Port ( R : in  STD_LOGIC;
           Clk : in  STD_LOGIC;
           D : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end DflipFLopReinAssync;

architecture Behavioral of DflipFLopReinAssync is

begin
process (R, Clk)
    begin
        if R = '0' then
            Q <= '0';
            Qn <= '1';

        elsif rising_edge(Clk) then
            Q <= D;
            Qn <= not D;
        end if;
    end process;

end Behavioral;
```

Test Bench pour D_Flip_Flop avec réinitialisations asynchrones

```
   --Inputs
   signal R : std_logic := '1';
   signal Clk : std_logic := '0';
   signal D : std_logic := '0';

   --Outputs
   signal Q : std_logic;
   signal Qn : std_logic;

   -- Clock period definitions
   constant Clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: DflipFLopReinAssync PORT MAP (
          R => R,
          Clk => Clk,
          D => D,
          Q => Q,
          Qn => Qn
        );
```

```
   Clk_process :process
   begin
      Clk <= '0';
      wait for Clk_period/2;
      Clk <= '1';
      wait for Clk_period/2;
   end process;


   stim_proc: process
   begin
      R <= '0';
        wait for 20 ns;

        R <= '1';
        D <= '0';
        wait for clk_period * 2;

        D <= '1';
        wait for clk_period * 2;

        D <= '0';
        wait for clk_period / 2;

        R <= '0';
        wait for clk_period;
      wait;
   end process;

END;
```



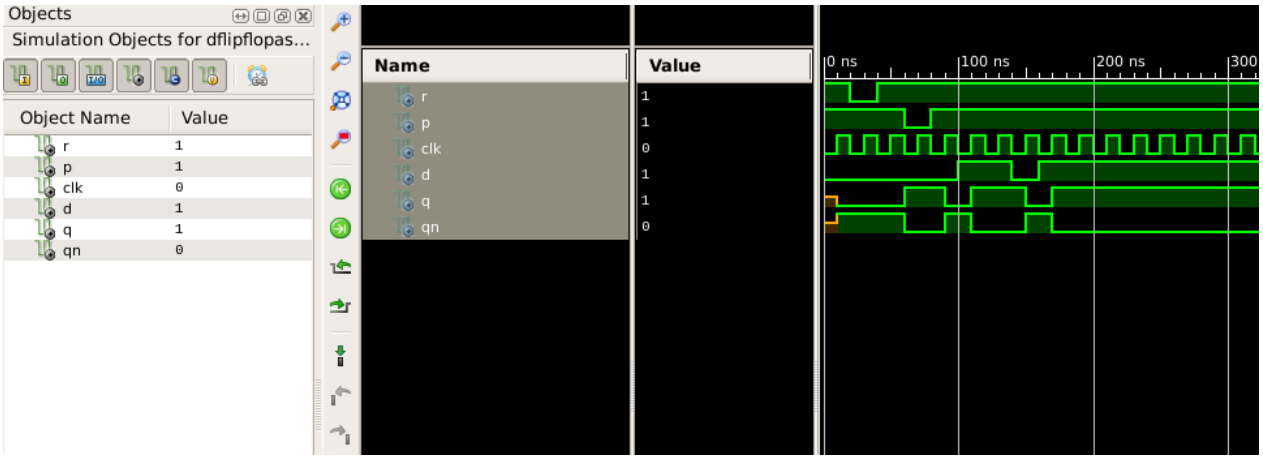
D_Flip_Flop avec Asynchrones Reset and Preset

```
31
32  entity dFLipFLopAsync is
33      Port ( R : in  STD_LOGIC;
34             P : in  STD_LOGIC;
35             Clk : in  STD_LOGIC;
36             D : in  STD_LOGIC;
37             Q : out  STD_LOGIC;
38             Qn : out  STD_LOGIC);
39  end dFLipFLopAsync;
40
41  architecture Behavioral of dFLipFLopAsync is
42
43  begin
44  process(R, P, Clk)
45      begin
46          if (R = '0') then
47              Q <= '0';
48              Qn <= '1';
49
50          elsif (P = '0') then
51              Q <= '1';
52              Qn <= '0';
53
54          elsif rising_edge(Clk) then
55              Q <= D;
56              Qn <= NOT D;
57          end if;
58      end process;
59
60  end Behavioral;
61
62
```

Test bench pour D_Flip_Flop avec Asynchrones Reset and Preset

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: dFLipFLopAsync PORT MAP (
          R => R,
          P => P,
          Clk => Clk,
          D => D,
          Q => Q,
          Qn => Qn
        );

   process
    begin
        while true loop
            Clk <= '0';
            wait for 10 ns;
            Clk <= '1';
            wait for 10 ns;
        end loop;
    end process;
```

```
 88
 89      process
 90      begin
 91          R <= '1'; P <= '1'; D <= '0'; wait for 20 ns;
 92
 93          R <= '0'; P <= '1'; wait for 20 ns;
 94          R <= '1'; wait for 20 ns;
 95
 96          R <= '1'; P <= '0'; wait for 20 ns;
 97          P <= '1'; wait for 20 ns;
 98
 99          R <= '1'; P <= '1'; D <= '1'; wait for 30 ns;
100
101          wait for 10 ns;
102          D <= '0'; wait for 20 ns;
103
104          D <= '1'; wait for 20 ns;
105
106          wait;
107      end process;
108
109  END;
110
```

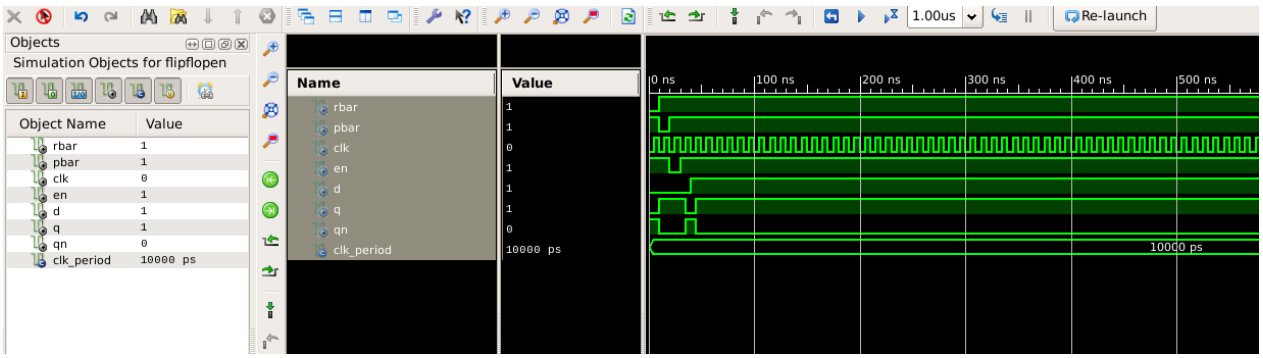


D_Flip_Flop avec Synchrone Enable

```
32  entity DFlipFlopSynchEn is
33      Port ( Rbar : in  STD_LOGIC;
34             Pbar : in  STD_LOGIC;
35             CLk : in  STD_LOGIC;
36             En : in  STD_LOGIC;
37             D : in  STD_LOGIC;
38             Q : out  STD_LOGIC;
39             Qn : out  STD_LOGIC);
40  end DFlipFlopSynchEn;
41
42  architecture Behavioral of DFlipFlopSynchEn is
43
44  begin
45  process (Rbar, Pbar, Clk)
46      begin
47          if Rbar = '0' then
48              Q <= '0';
49              Qn <= '1';
50          elsif Pbar = '0' then
51              Q <= '1';
52              Qn <= '0';
53          elsif rising_edge(Clk) then
54              if En = '1' then
55                  Q <= D;
56                  Qn <= not D;
57              end if;
58          end if;
59      end process;
60
61  end Behavioral;
62
63
```

Testbench pour D_Flip_Flop avec Synchrone Enable

```
82      -- Clock process definitions
83      CLk_process :process
84      begin
85          CLk <= '0';
86          wait for CLk_period/2;
87          CLk <= '1';
88          wait for CLk_period/2;
89      end process;
90
91
92      -- Stimulus process
93      stim_proc: process
94      begin
95           Rbar <= '0'; Pbar <= '1'; En <= '1'; D <= '0';
96           wait for clk_period;
97
98           Rbar <= '1'; Pbar <= '0';
99           wait for clk_period;
100
101          Rbar <= '1'; Pbar <= '1'; En <= '0';
102          wait for clk_period;
103
104          En <= '1'; D <= '0';
105          wait for clk_period;
106
107          D <= '1';
108          wait for clk_period;
109
110       wait;
111     end process;
112
113 END;
114
```



Exercice 2:

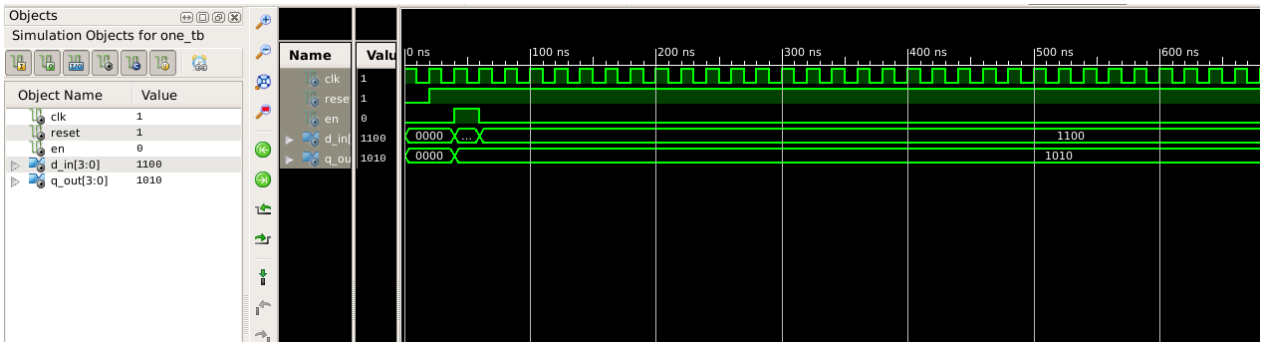
1) Register avec Enables 4 bits

Modèle VHDL avec generate

```
31
32  entity registerEnable is
33     Generic (
34          N : integer := 4
35      );
36      Port (
37          clk    : in  STD_LOGIC;
38          reset  : in  STD_LOGIC;
39          EN : in  STD_LOGIC;
40          D_in   : in  STD_LOGIC_VECTOR(N-1 downto 0);
41          Q_out  : out STD_LOGIC_VECTOR(N-1 downto 0)
42          );
43  end registerEnable;
44
45  architecture Behavioral of registerEnable is
46  signal reg : STD_LOGIC_VECTOR(N-1 downto 0);
47  begin
48  gen_reg: for i in 0 to N-1 generate
49          process(clk, reset)
50          begin
51              if reset = '0' then
52                  reg(i) <= '0';
53              elsif rising_edge(clk) then
54                  if EN = '1' then
55                      reg(i) <= D_in(i);
56                  end if;
57              end if;
58          end process;
59      end generate gen_reg;
60      Q_out <= reg;
61  end Behavioral;
62
```

Test bench:

```
72
73      clk_process: process
74       begin
75            while true loop
76                 clk <= not clk;
77                 wait for 10 ns;
78            end loop;
79       end process;
80
81      stim_proc: process
82      begin
83            reset <= '0';
84            wait for 20 ns;
85            reset <= '1';
86
87            wait for 20 ns;
88            EN <= '1';
89            D_in <= "1010";
90            wait for 20 ns;
91
92            en <= '0';
93            D_in <= "1100";
94            wait for 20 ns;
95
96          wait;
97      end process;
98
99  END;
100
```



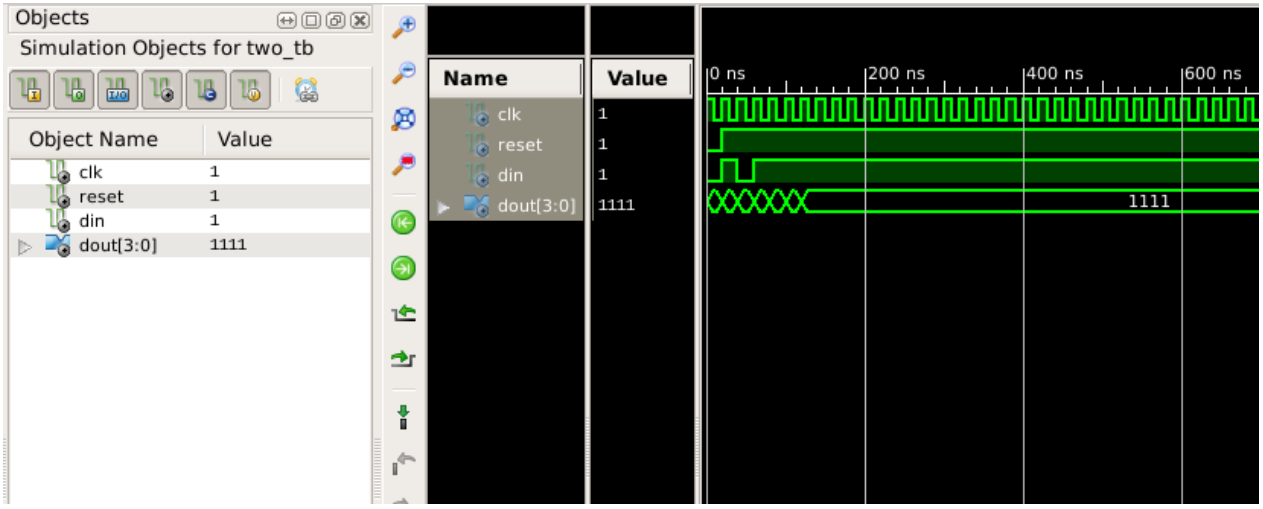
2) Shift Registers:

Modèle VHDL:

```
31
32  entity ShiftRegister is
33      Port ( clk : in  STD_LOGIC;
34             reset : in  STD_LOGIC;
35             Din : in  STD_LOGIC;
36             Dout : out  STD_LOGIC_VECTOR(3 downto 0)
37             );
38  end ShiftRegister;
39
40  architecture Behavioral of ShiftRegister is
41  signal reg : STD_LOGIC_VECTOR(3 downto 0);
42  begin
43  process(clk, reset)
44      begin
45          if reset = '0' then
46              reg <= (others => '0');
47              elsif rising_edge(clk) then
48              reg <= Din & reg(3 downto 1);
49          end if;
50      end process;
51  Dout <= reg;
52  end Behavioral;
53
54
```

Test Bench:

```
71
72      clk_process: process
73       begin
74           while true loop
75                clk <= not clk;
76                wait for 10 ns;
77           end loop;
78       end process;
79
80
81      -- Stimulus process
82      stim_proc: process
83      begin
84         reset <= '0';
85         wait for 20 ns;
86         reset <= '1';
87
88         Din <= '1';
89         wait for 20 ns;
90         Din <= '0';
91         wait for 20 ns;
92         Din <= '1';
93         wait for 20 ns;
94         Din <= '1';
95         wait for 20 ns;
96
97         wait;
98      end process;
99
100 END;
```



3) Registres sur un Bus de données

Modèle VHDL :

```
entity bus_registers is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           A_EN : in  STD_LOGIC;
           B_EN : in  STD_LOGIC;
           C_EN : in  STD_LOGIC;
           Din : in  STD_LOGIC_VECTOR (7 downto 0);
           A : out  STD_LOGIC_VECTOR (7 downto 0);
           B : out  STD_LOGIC_VECTOR (7 downto 0);
           C : out  STD_LOGIC_VECTOR (7 downto 0));
end bus_registers;

architecture Behavioral of bus_registers is
signal regA : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal regB : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal regC : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
begin
process(clk, reset)
    begin
        if reset = '1' then
            regA <= (others => '0');
            regB <= (others => '0');
            regC <= (others => '0');
        elsif rising_edge(clk) then
            if A_EN = '1' then
                regA <= Din;
            end if;
            if B_EN = '1' then
                regB <= Din;
            end if;
            if C_EN = '1' then
                regC <= Din;
            end if;
        end if;
    end process;

   A <= regA;
   B <= regB;
   C <= regC;

end Behavioral;
```

Test Bench :

```
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY THREE_TB IS
36  END THREE_TB;
37
38  ARCHITECTURE behavior OF THREE_TB IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT bus_registers
43      PORT(
44           clk : IN  std_logic;
45           reset : IN  std_logic;
46           A_EN : IN  std_logic;
47           B_EN : IN  std_logic;
48           C_EN : IN  std_logic;
49           Din : IN  std_logic_vector(7 downto 0);
50           A : OUT  std_logic_vector(7 downto 0);
51           B : OUT  std_logic_vector(7 downto 0);
52           C : OUT  std_logic_vector(7 downto 0)
53          );
54      END COMPONENT;
57      --Inputs
58      signal clk : std_logic := '0';
59      signal reset : std_logic := '0';
60      signal A_EN : std_logic := '0';
61      signal B_EN : std_logic := '0';
62      signal C_EN : std_logic := '0';
63      signal Din : std_logic_vector(7 downto 0) := (others => '0');
64
65      --Outputs
66      signal A : std_logic_vector(7 downto 0);
67      signal B : std_logic_vector(7 downto 0);
68      signal C : std_logic_vector(7 downto 0);
69
70
71  BEGIN
72
73      -- Instantiate the Unit Under Test (UUT)
74      uut: bus_registers PORT MAP (
75              clk => clk,
76              reset => reset,
77              A_EN => A_EN,
78              B_EN => B_EN,
79              C_EN => C_EN,
80              Din => Din,
81              A => A,
82              B => B,
83              C => C
84            );
85
```

```
85
86      clk_process: process
87       begin
88            while true loop
89                 clk <= not clk;
90                 wait for 10 ns;
91            end loop;
92       end process;
93
94      -- Stimulus process
95      stim_proc: process
96      begin
97      reset <= '1';
98      wait for 20 ns;
99      reset <= '0';
100
101     Din <= "10101010"; A_EN <= '1'; wait for 20 ns;
102          A_EN <= '0';
103     Din <= "11001100"; B_EN <= '1'; wait for 20 ns;
104          B_EN <= '0';
105     Din <= "11110000"; C_EN <= '1'; wait for 20 ns;
106          C_EN <= '0';
107        wait;
108     end process;
109
110 END;
```