Exercices de synthèse

Solution ALU

- Le code VHDL de l'ALU

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity alu is
generic (N: integer:=8);
port(
     a,b       : in  std_logic_vector(N-1 downto 0);
     operation: in  std_logic_vector(2 downto 0);
     res       : out  std_logic_vector(N downto 0));
end;

architecture archConc of alu is
begin
with operation select res <=
       std_logic_vector(unsigned('0' & a)+unsigned('0' & b)) when "000",
       std_logic_vector(unsigned('0' & a)-unsigned('0' & b)) when "001",
       '0' & a and '0' & b when "010",
```

Exercices de synthèse

Solution ALU

```
        '0' & a or '0' & b  when "011",
        '0' & a xor '0' & b when "100",
        (others => '0') when others;
end archConc;
```

Exercices de synthèse

Solution ALU

- Le code d'un testbench de l'ALU (non-exhaustif)

```
library ieee;
use ieee.std_logic_1164.all;
entity alu_tb is
end;

architecture archi_alu_tb of alu_tb is
component alu
generic (N: integer:=8);
port(
     a,b       : in  std_logic_vector(N-1 downto 0);
     operation: in  std_logic_vector(2 downto 0);
     res       : out  std_logic_vector(N downto 0));
end component;
signal a_t,b_t: std_logic_vector(7 downto 0);
signal res_t  : std_logic_vector(8 downto 0);
signal op_t   : std_logic_vector(2 downto 0);
begin
```

Exercices de synthèse

Solution ALU

```
alu_1: alu generic map(8) port map(a_t,b_t,op_t,res_t);

op_t <= "000", "001" after 40 ns, "010" after 80 ns, "011" after 120
     ns, "100" after 160 ns;
a_t  <= x"01", x"02" after 10 ns, x"03" after 20 ns, x"04" after 30 ns,
        x"05" after 40 ns, x"06" after 50 ns, x"07" after 60 ns, x"08"
     after 70 ns,
        x"09" after 80 ns, x"0A" after 90 ns, x"0B" after 100 ns, x"0C"
     after 110 ns,
        x"0D" after 120 ns, x"0E" after 130 ns, x"0F" after 140 ns,
     x"10" after 150 ns;

b_t  <= x"10", x"0F" after 10 ns, x"0E" after 20 ns, x"0D" after 30 ns,
        x"0C" after 40 ns, x"0B" after 50 ns, x"0A" after 60 ns, x"09"
     after 70 ns,
        x"08" after 80 ns, x"07" after 90 ns, x"06" after 100 ns, x"05"
     after 110 ns,
        x"04" after 120 ns, x"03" after 130 ns, x"02" after 140 ns,
     x"01" after 150 ns;
```

Exercices de synthèse

Solution ALU

```
end;
```

Exercices de synthèse

Solution ALU

- Le code d'un testbench de l'ALU exhaustif sans assertions

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity alu_tb_full is
end;

architecture archi_alu_tb_full of alu_tb_full is
component alu
generic (N: integer:=5);
port(
     a,b       : in  std_logic_vector(N-1 downto 0);
     operation: in  std_logic_vector(2 downto 0);
     res       : out  std_logic_vector(N downto 0));
end component;
signal a_t,b_t: std_logic_vector(4 downto 0);
signal res_t  : std_logic_vector(5 downto 0);
signal op_t   : std_logic_vector(2 downto 0);
```

Exercices de synthèse

Solution ALU

```
signal cntA,cntB: integer range 0 to 32;
signal cntOp: integer range 0 to 7;
begin
alu_1: alu generic map(5) port map(a_t,b_t,op_t,res_t);

process
begin
   wait for 10 ns;
   cntA <= cntA+1;
   if cntA=31 then
      cntA<=0;
      cntB <= CntB+1;
      if cntB=31 then
               cntB<=0;
         cntOp <= cntOp+1;
         if cntOp=4 then
            cntOp <= 0;
         end if;
      end if;
```

Exercices de synthèse

Solution ALU

```
   end if;
end process;

a_t <= std_logic_vector(to_unsigned(cntA,5));
b_t <= std_logic_vector(to_unsigned(cntB,5));
op_t <= std_logic_vector(to_unsigned(cntOp,3));

end;
```

Exercices de synthèse

Solution ALU

- Le code d'un testbench de l'ALU exhaustif (avec `for loop`) sans assertions

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity alu_tb_full is
end;

architecture archi_alu_tb_full of alu_tb_full is
component alu
generic (N: integer:=5);
port(
     a,b       : in  std_logic_vector(N-1 downto 0);
     operation: in  std_logic_vector(2 downto 0);
     res       : out  std_logic_vector(N downto 0));
end component;
signal a_t,b_t: std_logic_vector(4 downto 0);
signal res_t  : std_logic_vector(5 downto 0);
signal op_t   : std_logic_vector(2 downto 0);
```

Exercices de synthèse

Solution ALU

```
begin
alu_1: alu generic map(5) port map(a_t,b_t,op_t,res_t);
process
   variable cntA,cntB: integer range 0 to 2**5-1;
   variable cntOp: integer range 0 to 4;
begin
   for cntOp in 0 to 4 loop
      op_t <= std_logic_vector(to_unsigned(cntOp,3));
      for cntA in 0 to 31 loop
         a_t <= std_logic_vector(to_unsigned(cntA,5));
               for cntB in 0 to 31 loop
                  b_t <= std_logic_vector(to_unsigned(cntB,5));
            wait for 10 ns;
         end loop;
      end loop;
   end loop;
end process;
end;
```

Exercices de synthèse

Solution ALU

- Le code d'un testbench de l'ALU exhaustif (avec `for loop`) avec assertions

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity alu_tb_full is
end;

architecture archi_alu_tb_full of alu_tb_full is
component alu
generic (N: integer:=5);
port(
     a,b       : in  std_logic_vector(N-1 downto 0);
     operation: in  std_logic_vector(2 downto 0);
     res       : out  std_logic_vector(N downto 0));
end component;
signal a_t,b_t: std_logic_vector(4 downto 0);
signal res_t  : std_logic_vector(5 downto 0);
signal op_t   : std_logic_vector(2 downto 0);
```

Exercices de synthèse

Solution ALU

```
begin
alu_1: alu generic map(5) port map(a_t,b_t,op_t,res_t);
process
   variable cntA,cntB: integer range 0 to 2**5-1;
   variable cntOp: integer range 0 to 4;
begin
   for cntOp in 0 to 4 loop
      op_t <= std_logic_vector(to_unsigned(cntOp,3));
      for cntA in 0 to 31 loop
         a_t <= std_logic_vector(to_unsigned(cntA,5));
               for cntB in 0 to 31 loop
                  b_t <= std_logic_vector(to_unsigned(cntB,5));
            wait for 10 ns;
            case (cntOp) is
               when 0 =>
                  assert
    res_t=std_logic_vector(unsigned('0'&a_t)+unsigned('0'&b_t)) report
    "Erreur: A="& integer'image(cntA) &
```

Exercices de synthèse

Solution ALU

```
                " B= " & integer'image(cntB) & " Operation= " &
integer'image(cntOp) &
                " Resultat: " & integer'image(cntA+cntB) severity
failure;
                   when 1 =>
                assert
res_t=std_logic_vector(unsigned('0'&a_t)-unsigned('0'&b_t)) report
"Erreur: A="& integer'image(cntA) &
                " B= " & integer'image(cntB) & " Operation= " &
integer'image(cntOp) &
                " Resultat: " & integer'image(cntA-cntB) severity
failure;
                   when 2 =>
                assert res_t=(('0'&a_t) and ('0'& b_t)) report
"Erreur: A="& integer'image(cntA) &
                " B= " & integer'image(cntB) & " Operation= " &
integer'image(cntOp) &
```

Exercices de synthèse

Solution ALU

```
                " Resultat: " &
integer'image(to_integer(to_unsigned(cntA,5) and
to_unsigned(cntB,5))) severity failure;
                    when 3 =>
                assert res_t=(('0'&a_t) or ('0'&b_t)) report "Erreur:
A="& integer'image(cntA) &
                " B= " & integer'image(cntB) & " Operation= " &
integer'image(cntOp) &
                " Resultat: " &
integer'image(to_integer(to_unsigned(cntA,5) or
to_unsigned(cntB,5))) severity failure;
                    when 4 =>
                assert res_t=(('0'&a_t) xor ('0'&b_t)) report
"Erreur: A="& integer'image(cntA) &
                " B= " & integer'image(cntB) & " Operation= " &
integer'image(cntOp) &
                " Resultat: " &
integer'image(to_integer(to_unsigned(cntA,5) xor
to_unsigned(cntB,5))) severity failure;
```

Exercices de synthèse

Solution ALU

```
            end case;
         end loop;
      end loop;
   end loop;
end process;

end;
```