C9 – Generic, generate

Yann DOUZE

Port map

```
-- entité du composant COUNTER
entity COUNTER is
  port ( CLK, RST : in      Std_logic;
         UpDn     : in      Std_logic :=‘0’;
         Q        : out     Std_logic_vector(2 downto 0));
end entity;
--Architecture STRUCT d’un composant BLOK instanciant COUNTER
architecture STRUCT of BLOK is
begin
  -- association par position
  G1 : entity work.COUNTER port map (Clk32MHz, RST, open, Count);
  -- association par nom
  G2 : entity work.COUNTER port map( RST  => RST,
                                     CLK  => Clk32MHz,
                                     Q(2) => Q1MHz,
                                     Q(1) => Q2MHz,
                                     Q(0) => Q4MHz);
end architecture;
```

Valeur par défaut

Non connecté

VHDL 93

Compteur 8 bits

```
entity COUNTER8BIT is
   port ( CLK, RST : in      Std_logic;
          Q        : out     Std_logic_vector( 7 downto 0));
end entity;
architecture RTL of COUNTER8BIT is
   signal CNT: unsigned( 7 downto 0);
begin
   process (CLK,RST)
   begin
      if RST = ‘1’ then
         CNT <= “00000000”;
      elsif rising_edge(CLK) then
         CNT <= CNT + 1;
      end if;
   end process;
   Q <= std_logic_vector(CNT);
end architecture;
```

Compteur génériques

```
entity COUNTER is
generic(N : integer:=8);
port ( CLK, RST : in      Std_logic;
          Q         : out    Std_logic_vector(N-1 downto 0));
end entity;
architecture RTL of COUNTER is
  signal CNT: unsigned(N-1 downto 0);
begin
  process (CLK,RST)
  begin
    if RST = '1' then
      CNT <= (others => '0');
    elsif rising_edge(CLK) then
      CNT <= CNT + '1';
    end if;
  end process;
  Q <= std_logic_vector(CNT);
end architecture;
```

Instanciation d’un composant générique

```
-- l’entité du composant COUNTER
entity COUNTER is
  generic(N : integer:=8);
  port (CLK, RST : in  Std_logic;
        Q        : out   Std_logic_vector(N-1 downto 0));
end entity;

-- Utilisé dans l’architecture STRUCT d’un composant BLOK
architecture STRUCT of BLOK is
  signal Count4: std_logic_vector(3 downto 0);
  signal Count6: std-logic_vector(5 downto 0);
begin
-- association par position
  U1: entity work.COUNTER generic map(4)
  port map(CLK , RST, Count4);
-- association par nom
  U2: entity work.COUNTER generic map (N => 6)
  port map (CLK => CLK, RST => RST, Q => Count6);
end architecture;
```

Les délais génériques

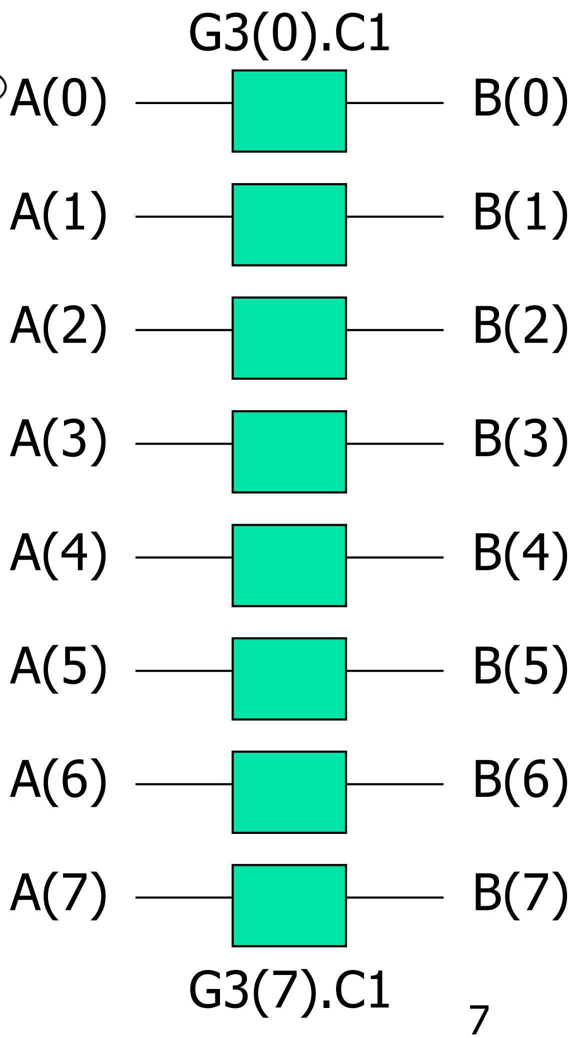
```
-- Entité du composant NAND2
entity NAND2 is
  generic (TPLH,TPHL: TIME := 0 NS);
  port ( A, B : in     Std_logic;
         F    : out    Std_logic);
end entity;

-- Architecture STRUCT du composant BLOK
architecture STRUCT of BLOK is
  signal N1,N2,N3,N4,N5,N6,N7,N8,N9 : Std_logic;
begin
  G1: entity work.NAND2 generic map (1.9 NS, 2.8 NS)
      port map (N1, N2, N3);
  G2: entity work.NAND2 generic map (TPLH => 2 NS, TPHL => 3 NS)
      port map (A => N4, B => N5, F => N6);
  G3: entity work.NAND2 port map (A => N7, B => N8, F => N9);
end architecture;
```

Instruction generate

```
architecture A1 of BLOK is
begin
 G1: for I in SOME_RANGE generate
     -- Instanciation de composant ou process
 end generate;

 G2: if CONDITION generate
     -- Instanciation de composant ou process
 end generate;

--Exemple :
 G3: for I in 0 to 7 generate
  C1: entity work.COMP port map (D=>A(I),Q=>B(I));
 end generate;

end architecture;
```

Structure régulière

Structure optionnelle



Additionneur structurel générique

```
entity ADDN is
  generic(N: positive :=4);
  port(  Cin : in std_logic;
         A,B: in std_logic_vector(N-1 downto 0);
         Cout : out std_logic;
         SUM: out std_logic_vector(N-1 downto 0));
end entity;
architecture STRUCT of ADDN is
signal C : std_logic_vector(N downto 0);
begin
  C(0) <= Cin;
  L1: for I in A’reverse_range generate
    U1 : entity work.ADDC1 port map(
    Cin => C(I), A => A(I), B => B(I),
    SUM => SUM(I), Cout => C(I+1));
  end generate;
  Cout <= C(N);
end architecture;
```