Numération - Codage des nombres-VHDL

Exercice 1 : numération - codage

1. Convertir en décimal les nombres suivants exprimés en binaire (entiers non signés) : $(0101)_2$; $(1111)_2$; $(110010)_2$; $(10010101)_2$
2. Ecrire en binaire les suites hexadécimales suivantes : $(73)_{16}$; $(232)_{16}$; $(3F8)_{16}$
3. Ecrire en hexadecimal les suites binaires suivantes : $(00100101)_2$; $(10101100)_2$; $(10101111111)_2$; $(0001000)_2$
4. Convertir en binaire et en hexadecimal (codage de nombres entiers non signés) les valeurs décimales suivantes : $(14)_{10}$; $(69)_{10}$; $(127)_{10}$; $(128)_{10}$; $(3762)_{10}$;
5. Généralisation : combien de bits (n)sont nécessaires pour coder en binaire (entier non signé) un nombre décimal (x) donné ? Etablir la formule mathématique qui permet de calculer cela systématiquement et très rapidement. Vérifier votre formule en testant pour $x = 15$, $x = 37$, $x = 35432$ et $x = 68364$.

Exercice 2 : numération... entiers signés et nombres à virgule

1. Après avoir étudier les 2 approches du codage des entiers signés, donner le codage en base 2 des nombres suivants : $(-7)_{10}$; $(-125)_{10}$
2. Donner ensuite la valeur décimale des entiers signés suivants : $(10111110)_2$; $(10000101)_2$
3. Etudier le principe de codage des nombres à virgule : virgule fixe et virgule flottante réel simple précision (float)
4. Donner le codage de $K = 0,607252935$ en virgule fixe $Q_{3.13}$ et en virgule flottante simple précision

Exercice 3 : VHDL - affectation et entité

1. En considèrant le schéma de la figure 1 ci-dessous, donner le codage VHDL de l'affectation des signaux de sortie.
2. En considérant le schéma de la figure 2 ci-dessous, donner le codage VHDL de l'entité de la fonction/application Fct.

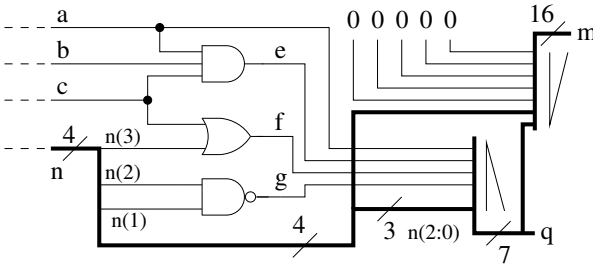

figure 1

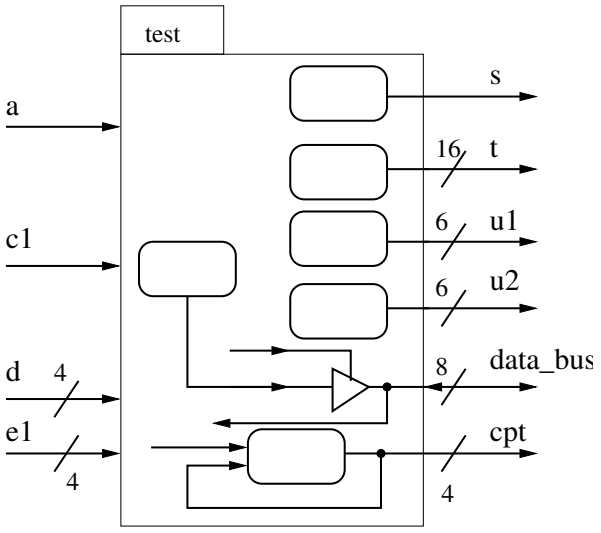

figure 2

Exercice 3 : VHDL - description hiérarchique port map vers schéma

1. On considère la description VHDL suivante. Dessiner le schéma logique hiérarchique équivalent

```
entity convertisseur is port (
    clk, en, init : in std_logic;
    entreebin8 : in std_logic_vector(7 downto 0);
    unite, dizaine : out std_logic_vector(3 downto 0));
end convertisseur;
architecture convert of convertisseur is
```

```
    -- declaration des component
    component CB8EDL is port (
       clk :in std_logic;
       en : in std_logic;
       load : in std_logic;
       e8 : in std_logic_vector(7 downto 0);
       done : out std_logic;
       s8: out std_logic_vector(7 downto 0)
    );
    end component;
    component cmpt_bcd10 IS PORT(
       clk, en, Init : IN std_logic;
       rco : OUT std_logic;
       q : OUT std_logic_vector(3 downto 0));
    END component;
    component sequenceur is port (
       clk,en,done,init : in std_logic;
       endec,load,memorize : out std_logic);
    end component;
    -- declaration des signaux internes
    signal en_seq,s_done,s_load,s_rco,s_memorize : std_logic;
    signal regAff : std_logic_vector(7 downto 0);
    signal s_dizaine, s_unite : std_logic_vector(3 downto 0);
begin
conv1:CB8EDL port map (
    clk => clk,en => en_seq,load=>s_load,e8 =>entreebin8,done => s_done,
    s8(3 downto 0) => open,s8(7 downto 4)=>open);
conv2:sequenceur port map (
    clk=>clk,en => en, endec => en_seq,load=>s_load,
    done => s_done,init => init,
    memorize=>s_memorize);
conv3:cmpt_bcd10 port map (
    clk => clk,en => en_seq, Init => s_load,rco => s_rco, q => s_unite);
conv4:cmpt_bcd10 port map (
    clk => clk,en => s_rco, Init => s_load,rco => open, q => s_dizaine);
-- registre d'affichage
process(clk) begin
    if rising_edge(clk) then
       if Init = '1' then
         regAff <= x"00";
       elsif s_memorize='1' then
         regAff <= s_dizaine & s_unite;
       end if;
    end if;
end process;
dizaine <= regAff(7 downto 4);
unite <= regAff(3 downto 0);
end convert;
```