

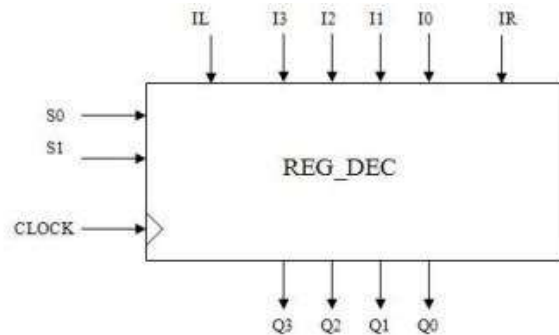
IV. l'architecture d'un Compteur 4 bits

Décrivez l'entité et l'architecture d'un Compteur 4 bits mise à un SET et mise à zéro RESET synchrone et l'entité et l'architecture d'un asynchrone.

<u>Compteur 4 bits synchrone:</u> <pre>Library ieee; Use ieee.std_logic_1164.all; Use ieee.numeric_std.all; entity CMP4BITS is port (CLK,SET,RESET : in std_logic; Q : inout std_logic_vector(3 downto 0)); end CMP4BITS; architecture DESCRIPTION of CMP4BITS is begin PRO_Count : process (CLK) Begin if (CLK'event and CLK='1') then if (RESET='1') then Q<= "0000"; elsif (SET='1')then Q <= "1111"; else Q <= Q+1; end if; end if; end process PRO_Count; end DESCRIPTION;</pre>	<u>Compteur 4 bits asynchrone:</u> <pre>Library ieee; Use ieee.std_logic_1164.all; Use ieee.numeric_std.all; entity CMP4BITS is port (CLK,SET,RESET : in std_logic; Q : inout std_logic_vector(3 downto 0)); end CMP4BITS; architecture DESCRIPTION of CMP4BITS is begin PRO_Count : process (CLK, RESET, SET) Begin if (RESET='1') then Q<= "0000"; elsif (SET='1')then Q <= "1111"; else if (CLK'event and CLK='1') then Q <= Q+1; end if; end process PRO_Count; end DESCRIPTION;</pre>
---	--

V. Registre à décalage

Le circuit d'un registre à décalage synchrone (front montant) est donné sur la figure suivante :



- 4 entrées/4 sorties
- 1 entrée horloge
- 2 entrées de sélection qui permettent de choisir le mode :

S0	S1	Fonction réalisée après le top horloge	Opération
0	0	$Q[3..0] \leq Q[3..0]$	Pas de changement
0	1	$Q[3..0] \leq I[3..0]$	Chargement parallèle
1	0	$Q[3..1] \leq Q[2..0], Q[0] \leq IR$	Décalage à gauche avec IR
1	1	$Q[3] \leq IL, Q[2..0] \leq Q[3..1]$	Décalage à droite avec IL

- 1) Ecrire l'entité qui décrit la vue externe de ce registre.
- 2) Donner une architecture comportementale (tableau ci-dessus) en utilisant un process. Dans le process, vous pouvez utiliser l'instruction if.... Then.....elsif et l'instruction for.....loop.

<pre> library ieee; use ieee.std_logic_1164.all; entity register1 is port (H, IR, IL, S0, S1 : in std_logic; I: in std_logic_vector (3 downto 0); Q : buffer std_logic_vector (3 downto 0)); end register1; architecture arch of register1 is Begin process (S0, S1, H) variable Z : integer; variable N: Integer:=0; begin if (H'event and H='1') then if (S0='0' and S1='0') then Q<=Q; elsif (S0='0' and S1='1') then Q<=I; </pre>	<pre> elsif (S0='1' and S1='0') then For Z IN 3 downto 1 Loop N:=Z-1; Q(Z)<=Q(N); End loop; Q(N)<=IR; elsif (S0='1' and S1='1') then For Z IN 0 to 2 Loop N:=Z+1; Q(Z)<=Q(N); End loop; Q(N)<=IL; end if; end if; end process; end arch; </pre>
--	---

IV. Décodeur binaire / 7 segments

Ecrire l'ensemble d'un fichier VHDL (Library, Entity, Architecture) qui décrit un décodeur binaire / 7 segments fig. 1, en utilisant l'instruction *case*.

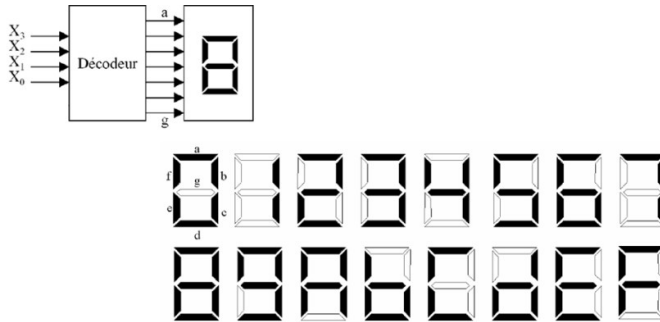


Fig. 1 Décodeur de type Hex avec afficheur 7 Segments

<pre> library ieee ; use ieee.std_logic_1164.ALL; entity HEX_7SEG is port (X: in STD_LOGIC_VECTOR(3 downto 0) ; a,b,c,d,e,f,g: out STD_LOGIC); end HEX_7SEG; architecture STRUCT of HEX_7SEG is signal S: STD_LOGIC_VECTOR(6 downto 0) ; begin process(X) begin -- SORTIE S= "abcdefg" case X is when 0 => S <= "1111110" ; when 1 => S <= "0110000" ; when 2 => S <= "1101101" ; . . . when 14 => S <= "1001111" ; when 15 => S <= "1000111" ; when others => S <= "01101101" ; -- X pour erreur end case ; </pre>	<pre> a<=S(6); b<=S(5); c<=S(4); d<=S(3); e<=S(2); f<=S(1); g<=S(0); end process ; end STRUCT; </pre>
--	--