***Treball previ 1 ---->***

Escriviu el bloc codnum en VHDL. Aquest ha de tenir l’entrada key de 14 bits i la sortida hex de 4 bits. Feu un disseny combinacional.

library ieee;

use ieee.std\_logic\_1164.all;

entity condum is

port(key: in std\_logic\_vector(13 down to 0);

hex: out in std\_logic\_vector(3 down to 0));

end codnum;

architecture arch of codnum is

signal x: std\_logic\_vector(3 down to 0);

x <= key(3 down to 0);

begin

if x = ’0000’ then hex <= ’0000’;

elsif x = ’0001’ then hex <= ’0001’

elsif x = ’0010’ then hex <= ’0010’

elsif x = ’0011’ then hex <= ’0011’

elsif x = ’0100’ then he x<= ’0100’

elsif x = ’0101’ then hex <= ’0101’

elsif x = ’0110’ then hex <= ’0110’

elsif x = ’0111’ then hex <= ’0111’

elsif x = ’1000’ then hex <= ‘1000’

elsif x = ’1001’ then hex <= ’1001’

else hex <= ’- - - -’

end if;

end arch;

***Treball previ 2 ---->***

Haureu observat que a les línies 16 a 29 hi apareixen uns interrogants. Es tracta de que esbrineu els valors que calen per aconseguir l’objectiu perseguit. Cerqueu la informació que us calgui i tingueu en compte que els noms dels ports s’han triat de forma racional.

library ieee;

use ieee.std\_logic\_1164.all;

entity hex7seg is

port(hex: in std\_logic\_vector(3 down to 0);

a,b,c,d,e,f,g,h: out std\_logic);

end hex7seg;

architecture arc of hex7seg is

signal bus\_out: std\_logic\_vector(6 down to 0);

begin

with hex select

bus\_out <= ‘1111110’ when x = ‘0’,

‘0110000’ when x = ‘1’,

‘1101101’ when x = ‘2’,

‘1111001’ when x = ‘3’,

‘1111001’ when x = ‘4’,

‘1111001’ when x = ‘5’,

‘1111001’ when x = ‘6’,

‘1111001’ when x = ‘7’,

‘1111001’ when x = ‘8’,

‘1111001’ when x = ‘9’,

‘- - - - - - -’ when others;

(a,b,c,d,e,f,g) <= bus\_out;

end arc;

***Treball previ 3 ---->***

Algú podria pensar que els dos darrers blocs es podrien combinar en un bloc únic. S’ha triat fer-ho de forma separada per millorar la claredat del disseny. D’altra banda, durant el procés de síntesi, l’optimitzador ja s’encarregarà de trobar la forma més simplificada de realitzar la funció final. Malgrat aquests comentaris, proposeu un únic bloc VHDL que substitueixi la funcionalitat dels dos darrers blocs.

library ieee;

use ieee.std\_logic\_1164.all;

entity condum is

port(key: in std\_logic\_vector(13 down to 0);

hex: out in std\_logic\_vector(3 down to 0));

end codnum;

architecture arch of codnum is

signal x: std\_logic\_vector(3 down to 0);

x <= key(3 down to 0);

begin

if x = ’0000’ then hex <= ’0000’;

elsif x = ’0001’ then hex <= ’0001’

elsif x = ’0010’ then hex <= ’0010’

elsif x = ’0011’ then hex <= ’0011’

elsif x = ’0100’ then he x<= ’0100’

elsif x = ’0101’ then hex <= ’0101’

elsif x = ’0110’ then hex <= ’0110’

elsif x = ’0111’ then hex <= ’0111’

elsif x = ’1000’ then hex <= ‘1000’

elsif x = ’1001’ then hex <= ’1001’

else hex <= ’----’

end if;

end arch;

entity hex7seg is

port(hex: in std\_logic\_vector(3 down to 0);

a,b,c,d,e,f,g,h: out std\_logic);

end hex7seg;

architecture arc of hex7seg is

signal bus\_out: std\_logic\_vector(6 down to 0);

begin

with hex select

bus\_out <= ‘1111110’ when x = ‘0’,

‘0110000’ when x = ‘1’,

‘1101101’ when x = ‘2’,

‘1111001’ when x = ‘3’,

‘1111001’ when x = ‘4’,

‘1111001’ when x = ‘5’,

‘1111001’ when x = ‘6’,

‘1111001’ when x = ‘7’,

‘1111001’ when x = ‘8’,

‘1111001’ when x = ‘9’,

‘- - - - - - -’ when others;

(a,b,c,d,e,f,g) <= bus\_out;

end arc;

***Treball previ 4 ---->***

Cada vegada que es rep una trama, aquesta s’emmagatzema en un registre de desplaçament. Expliqueu com es veu afectat el sistema a mesura que van entrant els 14 bits. Es produeix algun fenomen indesitjat a les sortides? En cas afirmatiu, proposeu una manera per solucionar-lo. Podeu fer-ne el disseny amb VHDL o bé blocs digitals.

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all:

entity ir\_cod is

port(data,data\_valid: in std\_logic;

key: out std\_logic\_vector(13 down to 0));

end ir\_cod;

architecture arch of ir\_cod is

signal q: std\_logic\_vector(13 downto 0)

:= (others => '0');

signal counter: unsigned(3 downto 0)

:= (others => '0');

begin

process(data\_valid)

begin

if rising\_edge(data\_valid) then

counter <= counter + 1;

q <= q(12 downto 0) & data;

if counter = 14 then

key <= q;

counter <= "0000";

end if;

end if;

end process;

end arch;