Nome e Apelido:_____ Assinatura:_____N. Mec.:_____

Grupo I

1. *[1 valor]* O seguinte código VHDL descreve um multiplexador 8:1, mas apresenta omissões e erros de sintaxe. Corrija-o: pode fazê-lo sobre o código fornecido; risque e escreva à frente o código correto e/ou acrescente o que falta.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Mux8_1 is
     port map(sel     : std_logic_vector(___ downto 0);
              dataIn  : std_logic_vector( 7  downto 0);
              dataOut : std_logic);
end Mux8_1;

architecture _______________ of _______________ is
begin
    dataOut => dataIn(0) when (sel = '000') else
               dataIn(1) when (sel = '001') else
               dataIn(2) when (sel = '010') else
               dataIn(3) when (sel = '011') else
               dataIn(4) when (sel = '100') else
               dataIn(5) when (sel = '101') else
               dataIn(6) when (sel = '110') else
               dataIn(7) when (sel = '111') else
               dataIn;
end BehavAssign;
```

2. *[1.5 valores]* Escreva o código de um multiplexador 8:1 mas agora baseado num *processo* em VHDL. Considere a mesma entidade da alínea anterior.

```
architecture BehavProc of Mux8_1 is
begin
    process(                    )
    begin









    end process;
end BehavProc;
```

3. Considere o código VHDL da entidade *top-level* **NewBlock**, que instancia dois multiplexadores 4:1 (entidade **Mux4_1**) e um multiplexador 2:1 (entidade **Mux2_1**).

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity NewBlock is
   port(SW   : in  std_logic_vector(7 downto 0);
        KEY  : in  std_logic_vector(2 downto 0);
        LEDR : out std_logic_vector(0 downto 0));
end NewBlock;

architecture Structural of NewBlock is
    signal out_mux : std_logic_vector(1 downto 0);
begin
    mux41_0: entity work.Mux4_1(BeahvMux4_1)
             port map(dataIn  => SW(3 downto 0),
                      sel     => KEY(1 downto 0),
                      dataOut => out_mux(0));

    mux41_1: entity work.Mux4_1(BeahvMux4_1)
             port map(dataIn  => SW(7 downto 4),
                      sel     => KEY(1 downto 0),
                      dataOut => out_mux(1));

    mux21  : entity work.Mux2_1(BeahvMux2_1)
             port map(dataIn  => out_mux,
                      sel     => KEY(2),
                      dataOut => LEDR(0));
end Structural;
```

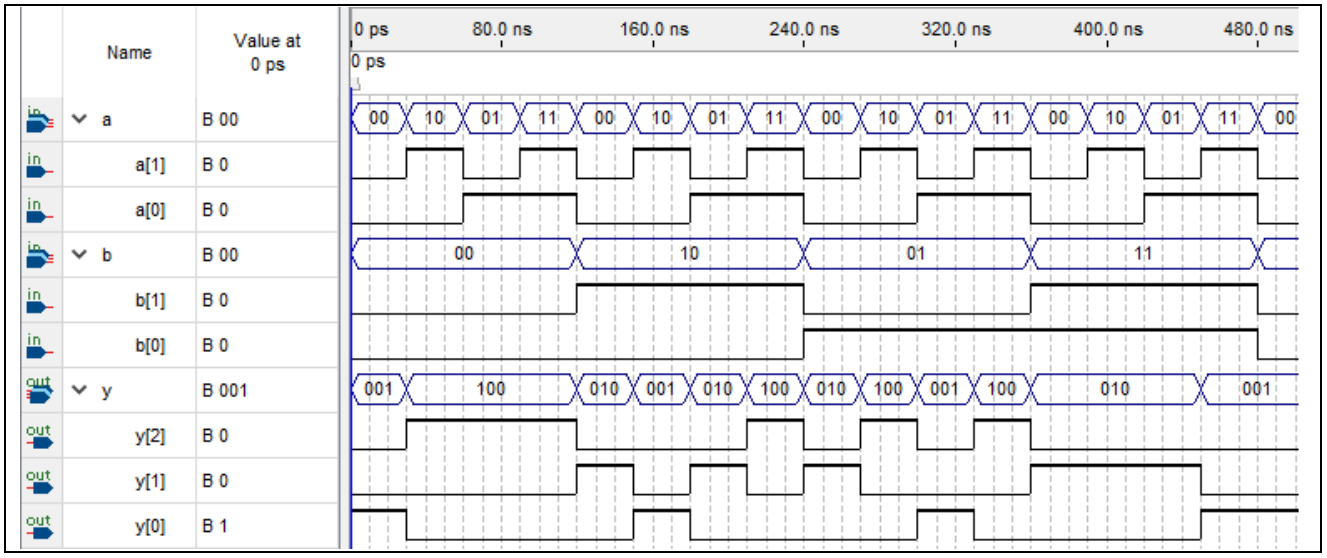
a) *[2 valores]* Com base no código fornecido, desenhe o diagrama esquemático do **NewBlock**. Identifique todos os sinais (nomes e dimensões) de entrada/saída de **NewBlock** e todos os sub-módulos.

b) *[0.5 valores]* Identifique a função de **NewBlock** e indique o valor lógico de **LEDR[0]**, considerando que **SW[7..0]="10110101"** e que **KEY[2..0]="110"**. **Justifique a sua resposta**.

Nome e Apelido:_____ Assinatura:_____N. Mec.:_____

Grupo II

Considere o seguinte diagrama temporal, resultante da simulação de um comparador com entradas **`a[1..0]`** e **`b[1..0]`** e saídas **`y[2..0]`** que disponibilizam os resultados das comparações **"<"**, **"="** e **">"**.



1. *[1 valor]* Relativamente a este diagrama temporal, qual o período do sinal de entrada que comuta com maior frequência? Qual o valor da frequência? Identifique o sinal e apresente os cálculos que efetuar.

2. *[1 valor]* Com base no diagrama temporal apresentado, identifique, **justificando**, a função de cada uma das saídas.

3. *[1 valor]* Com base no diagrama temporal apresentado, indique se as comparações são realizadas com sinal (*signed*) ou sem sinal (*unsigned*). **Justifique a sua resposta**.

4. *[2 valores]* Complete o código VHDL que descreve o comparador capaz de produzir o diagrama temporal anterior. Use atribuições condicionais **`when .. else`** para especificar cada uma das saídas.

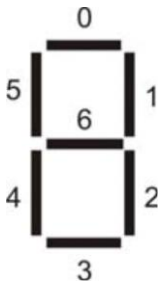
```
library IEEE;

use _____________________________________________________________________

use _____________________________________________________________________

entity Cmp2 is
    port(a : in std_logic_vector(1 downto 0);

         _____________________________________________________________________

         _____________________________________________________________________

end Cmp2;

architecture _____________________________________________________________

begin
    y(0) <= _____________________________________________________________

    y(1) <= _____________________________________________________________

    y(2) <= _____________________________________________________________

end _____________________________________________________________________
```

Área de rascunho

Nome e Apelido:_____ Assinatura:_____N. Mec.:_____

Grupo III

Pretende-se conceber um circuito que mostre em decimal duas interpretações possíveis de um conjunto de 4 *bits*: como um inteiro sem sinal (valores entre 0 e 15) ou como um inteiro com sinal (valores entre -8 e 7). Por exemplo, o vetor "1100" pode ser mostrado como 12 ou -4. Para esse fim, foram desenvolvidos 3 módulos, cujos objetivos e esqueleto em VHDL são apresentados nas questões seguintes. Em todas as questões deverá completar o respetivo código VHDL.



1. *[1.5 valores]* O módulo **`BCDDisplayDec`** apresenta num *display* de 7 segmentos (como o ilustrado ao lado) **valores positivos entre 0 e 9**, ou um **sinal negativo** (apenas o segmento 6 aceso) ou **nada** (todos os segmentos apagados).

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity BCDDisplayDec is
    port(signIn     : _____________________________________________________

         numberIn   : _____________________________________________________

         displayOut : _____________________________________________________);
end BCDDisplayDec;

architecture Behavioral of BCDDisplayDec is
begin
    process(_____________________________________)
    begin
        if (signIn = '1') then
            displayOut <= "_____________________";

        elsif (numberIn = "0000") then
            displayOut <= "_____________________";
        elsif (numberIn = "0001") then
            displayOut <= "1111001";
        elsif (numberIn = "0010") then
            displayOut <= "0100100";
        elsif (numberIn = "0011") then
            displayOut <= "_____________________";
        elsif (numberIn = "0100") then
            displayOut <= "0011001";
        elsif (numberIn = "0101") then
            displayOut <= "0010010";
        elsif (numberIn = "0110") then
            displayOut <= "0000011";
        elsif (numberIn = "0111") then
            displayOut <= "1111000";
        elsif (numberIn = "1000") then
            displayOut <= "_____________________";
        elsif (numberIn = "1001") then
            displayOut <= "0011000";
        else
            displayOut <= "1111111";
        end if;
    end process;
end Behavioral;
```

2. *[1.5 valores]* O módulo **`Conv4Bit2BCD`** converte para BCD um valor de 4 *bits* como um número sem sinal (binário natural) ou como um número com sinal (interpretado em complemento para 2).

```
library IEEE; use IEEE.STD_LOGIC_1164.all; use IEEE.NUMERIC_STD.all;

entity Conv4Bit2BCD is
    port(isSigned    : _____________________________________ -- 0: unsigned integer
                                                             -- 1: signed integer
         numberIn    : _____________________________________ -- binary input value
         lsBCDDigOut : _____________________________________ -- least significant digit
         msBCDDigOut : _____________________________________ -- most significant digit
         signOut     : _____________________________________);-- minus sign
end Conv4Bit2BCD;

architecture Behavioral of Conv4Bit2BCD is
begin
    process(_____________________________________________) is
    begin
        if (isSigned = '1') then         -- abs means absolute value
            lsBCDDigOut <= std_logic_vector(abs(signed(numberIn)));
            msBCDDigOut <= "1111";
            signOut     <= ____________________________________________________
        else
            lsBCDDigOut <= ____________________________________________________
            msBCDDigOut <= ____________________________________________________
            signOut     <= '0';
        end if;
    end process;
end Behavioral;
```

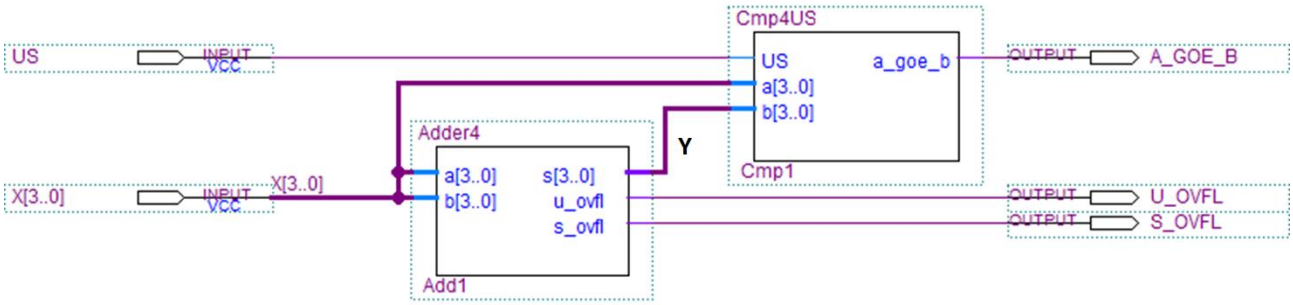
3. *[2 valores]* O módulo **`DisplaySystem`** apresenta um valor de 4 *bits*, indicado através dos interruptores **`SW`**, em dois *displays* de 7 segmentos. Quando o interruptor de pressão **`KEY0`** é pressionado, o valor de 4 *bits* é interpretado como um inteiro sem sinal, caso contrário é interpretado como um inteiro com sinal (complemento para 2). Declare os sinais que entender necessários.

```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity DisplaySystem is
    port(KEY  : ____________________________________________________________
         SW   : ____________________________________________________________
         HEX0 : ____________________________________________________________
         HEX1 : ____________________________________________________________);
end DisplaySystem;

architecture Structural of DisplaySystem is



begin
    bin_bcd_decoder : entity work.Conv4Bit2BCD(Behavioral)
        port map(isSigned    => ____________________________________________
                 numberIn    => ____________________________________________
                 lsBCDDigOut => ____________________________________________
                 msBCDDigOut => ____________________________________________
                 signOut     => ____________________________________________);

    display_dec_0 : entity work.BCDDisplayDec(Behavioral)
        port map(signIn     => ____________________________________________
                 numberIn   => ____________________________________________
                 displayOut => ____________________________________________);

    display_dec_1 : entity work.BCDDisplayDec(Behavioral)
        port map(signIn     => ____________________________________________
                 numberIn   => ____________________________________________
                 displayOut => ____________________________________________);
end Structural;
```

Nome e Apelido:_____ Assinatura:_____N. Mec.:_____

Grupo IV

Analise o sistema digital apresentado na figura seguinte.



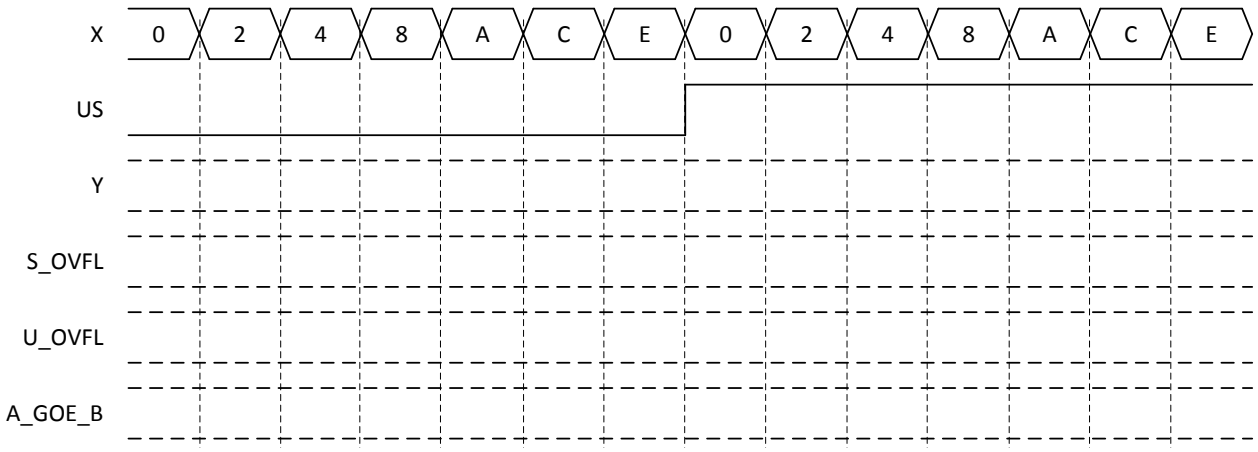
`Adder4` é um somador de 4 *bits*. **`u_ovfl`** e **`s_ovfl`** são indicadores de *overflow* (activos ao nível '1'):

- **`u_ovfl`** assume que os valores nas entradas **`a`** e **`b`** são números sem sinal (binário natural);
- **`s_ovfl`** assume que os valores nas entradas **`a`** e **`b`** são números com sinal (representação em complemento para 2).

`Cmp4US` é um comparador de 4 *bits* cuja saída **`a_goe_b`** fica activa (i.e. com valor lógico '1') quando o valor na entrada **`a`** é maior do que ou igual ao valor na entrada **`b`** *(a greater than or equal to b).* A entrada **`US`** especifica o modo de comparação:

- com **`US='0'`** os valores nas entradas **`a`** e **`b`** serão encarados como números sem sinal (binário natural);
- com **`US='1'`** os valores nas entradas **`a`** e **`b`** serão encarados como números com sinal (representação em complemento para 2).

1. *[2.5 valores]* O diagrama seguinte mostra uma sequência de vectores de teste aplicados às entradas do sistema (**`X[3..0]`** e **`US`**). Desenhe as correspondentes formas de onda das saídas (**`Y`**, **`S_OVFL`**, **`U_OVFL`** e **`A_GOE_B`**). À semelhança de **`X`**, os valores de **`Y`** devem ser indicados em hexadecimal.



2. *[2.5 valores]* Complete o seguinte código VHDL para **`Adder4`**. Lembre que, na adição de números com sinal, ocorre *overflow* se e só se os transportes (*carry*) de e para o *bit* mais significativo (*MSbit*) não forem iguais. Comece por acrescentar as instruções para obter o *carry* do *MSbit* (o *carry* para o *MSbit* já está contemplado). Declare os sinais adicionais que entender necessários. Complete depois o código referente às saídas.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity Adder4 is
    port(a, b   : in  std_logic_vector(3 downto 0);
         s      : out std_logic_vector(3 downto 0);
         u_ovfl : out std_logic;
         s_ovfl : out std_logic);
end Adder4;

architecture Behavioral of Adder4 is

    signal ta4, tb4, ts4       : unsigned(3 downto 0);

    signal MSbitCin, MSbitCout : _____________________________________________




begin
    ta4       <= '0' & unsigned(a(2 downto 0));
    tb4       <= '0' & unsigned(b(2 downto 0));
    ts4       <= ta4 + tb4;

    MSbitCin  <= ts4(3);                    -- carry para o Most Significant bit

    MSbitCout <=                           ; -- carry do Most Significant bit

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

    ____________________________________________________________________________

end Behavioral;
```