

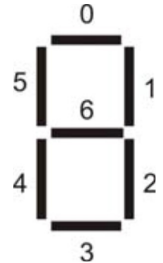
Nome: \_\_\_\_\_ N. Mec.: \_\_\_\_\_ Turma: \_\_\_\_\_

### Grupo I

1. [4 valores] No código seguinte, referente a um decodificador de binário (“0000” a “1111”) para um visualizador (*display*) de 7 segmentos, a entidade possui os seguintes portos:

- **enable\_n** é uma entrada de habilitação em lógica negativa (ativa baixa);
- **binInput** é a entrada de dados;
- **decOut** é uma saída em lógica positiva (ativa alta), cujos bits ligam aos segmentos de acordo com a figura do lado.

O código está incompleto e contém erros de sintaxe e funcionais. Corrija-o, assinalando os erros ou omissões com um círculo em redor do ponto onde ocorrem e escrevendo, na zona livre à direita, as correções ou adições que propõe.



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

entity Bin7SegDecoder is
    signal(enable_n : std_logic;
           binInput  : std_logic(3 downto 0);
           decOut    : std_logic(7 downto 0));
end Behavioral;

architecture Behavioral of Bin7SegDecoder is
begin
    decOut =>
        "0000000" when (enable_n = "0") else
        "0000110" when (binInput = "0001") else --1
        "    " when (binInput = "0010") else --2
        "1001111" when (binInput = "0011") else --3
        "1100110" when (binInput = "0100") else --4
        "1101101" when (binInput = "0101") else --5
        "1111101" when (binInput = "0110") else --6
        "0000111" when (binInput = "0111") else --7
        "1111111" when (binInput = "1000") else --8
        "1101111" when (binInput = "1001") else --9
        "1110111" when (binInput = "1010") else --A
        "1111100" when (binInput = "1011") else --B
        "0111001" when (binInput = "1100") else --C
        "1011110" when (binInput = "1101") else --D
        "1111001" when (binInput = "1110") else --E
        "1110001" when (binInput = "1111"); --F
end process;
```

## Grupo II

Pretende-se que construa um somador de 5 *bits* com detecção de *overflow* (módulo *Adder5Ovf*). Para o efeito, dispõe dos componentes *AdderN* e *FullAdder*, cujas interfaces se encontram definidas nos seguintes trechos de código VHDL.

```
entity AdderN is -- Somador parametrizável de N bits
  generic(N : positive:=6);
  port(a, b : in  std_logic_vector(N-1 downto 0);
        cin : in  std_logic;
        s   : out std_logic_vector(N-1 downto 0);
        cout : out std_logic);
end AdderN;
```

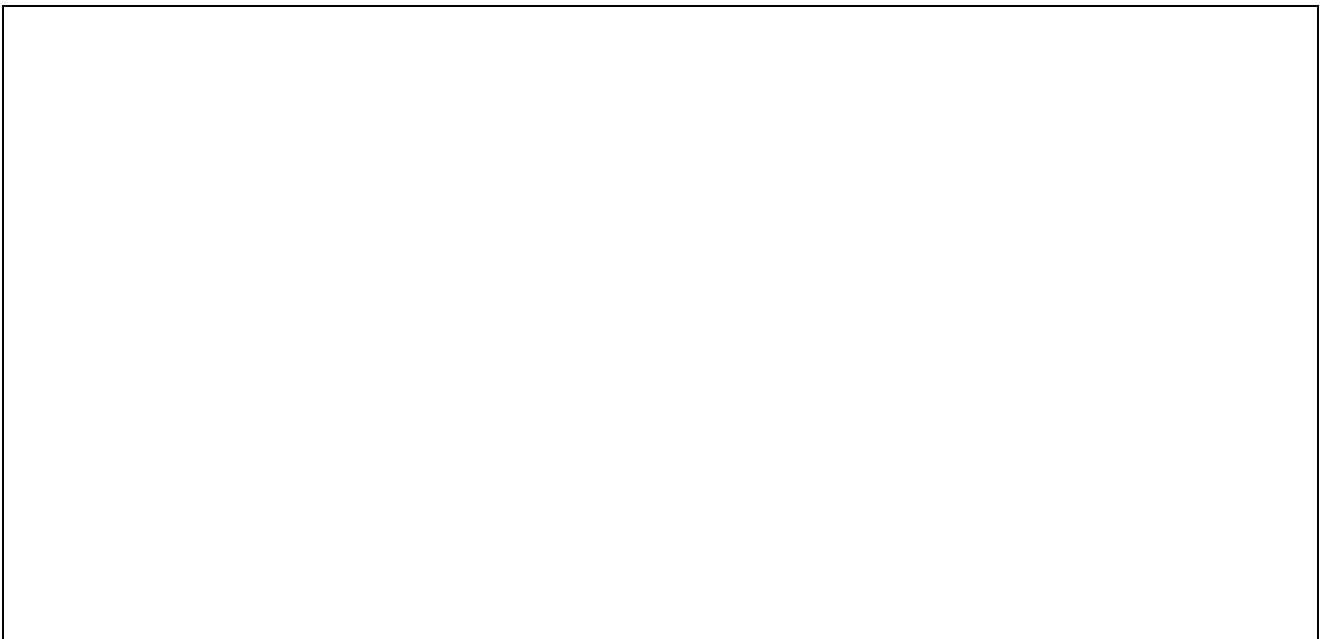
```
entity FullAdder is -- Somador completo de 1 bit
  port(a, b : in  std_logic;
        cin : in  std_logic;
        s   : out std_logic;
        cout : out std_logic); -- saída de transporte (carry out)
end FullAdder;
```

Considere também um esboço (incompleto) da entidade *Adder5Ovf*. Além dos portos relativos aos operandos (**a**, **b** in **cin**) e resultado (**s**), devem ser disponibilizadas duas saídas de detecção de *overflow*: **uovf** (*unsigned overflow*) e **sovf** (*signed overflow*). A saída **uovf** refere-se ao *overflow* na adição de números sem sinal, que ocorre quando o "carry out" do bit mais significativo do resultado é '1'. A saída **sovf** refere-se ao *overflow* na adição de números com sinal (representação em complemento para 2) e que ocorre quando o "carry in" e o "carry out" do bit mais significativo do resultado possuem valores lógicos opostos.

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

entity Adder5Ovf is -- Somador de 5 bits
  port(a, b :                                -- operandos
        cin :                                -- entrada de transporte (carry in)
        s   :                                -- soma
        sovf :                               -- flag de signed overflow
        uovf :                               -- flag de unsigned overflow
        );
end Adder5Ovf;
```

1. [2.5 valores] Apresente, através de um diagrama de blocos, a estrutura do módulo *Adder5Ovf* com base nos componentes *AdderN* e *FullAdder*, explicitando todos os portos e sinais de *Adder5Ovf* e suas interligações aos portos dos componentes instanciados. Minimize o número de componentes usados.



2. [3 valores] Complete agora o código VHDL apresentado para o módulo *Adder5Ovf*: complete a definição dos portos da entidade esboçada acima e preencha o código da arquitetura seguinte, de acordo com a estrutura que esquematizou no ponto anterior.

```
architecture Structural of Adder5Ovf is
```

```
end Structural;
```

3. [1.5 valores] Complete a seguinte tabela, indicando que valores assumirão as saídas para cada conjunto de valores de entrada.

a	b	cin	s	uovf	sovf
11001	11010	1			
01001	01000	0			

4. [2 valores] Complete agora o código seguinte do módulo de *top-level* *Adder5Demo* para efeitos de teste do componente *Adder5Ovf* no kit DE2-115. Use os LEDR apenas para sinalizar as situações de *overflow*.

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

```
entity Adder5Demo is
  port(SW    :
        LEDR :
        LEDG :                );
end Adder5Demo;
```

```
architecture Shell of Adder5Demo is
begin
  adder : entity work.Adder5Ovf(Structural)
    port map(
```

```
);
```

```
end Shell;
```

### Grupo III

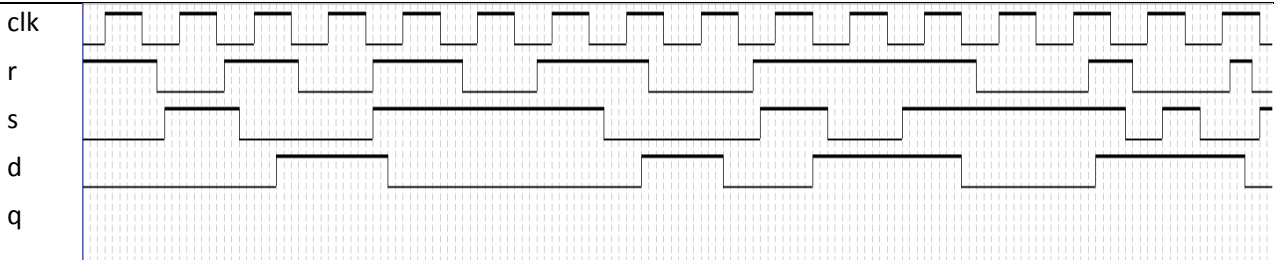
1. [3 valores] Analise atentamente o seguinte código VHDL e complete, com a representação do sinal de saída **q**, o diagrama temporal de uma simulação efetuada com a entidade **FlipFlopD** e com a arquitetura **Behav\_1** apresentadas. Preencha também adequadamente com “✓” a tabela seguinte.

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

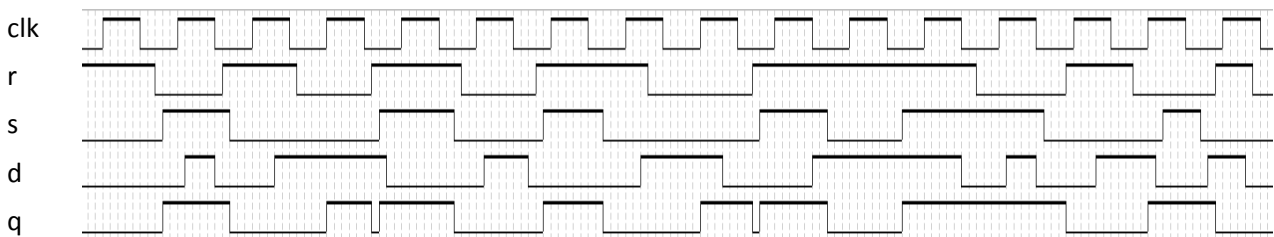
entity FlipFlopD is
    port(clk, r, s, d : in std_logic;
          q : out std_logic);
end FlipFlopD;

architecture Behav_1 of FlipFlopD is
begin
    process(clk, r)
    begin
        if (r = '1') then
            q <= '0';
        elsif (rising_edge(clk)) then
            if (s = '1') then
                q <= '1';
            else
                q <= d;
            end if;
        end if;
    end process;
end Behav_1;
```

Porto	Síncrono?	Assíncrono?
r (reset)		
s (set)		



2. [4 valores] A simulação seguinte usou a mesma entidade (**FlipFlopD**), mas a arquitetura **Behav\_2**, em vez da **Behav\_1**. Após perceber as diferenças de comportamento temporal, escreva o respetivo código da arquitetura **Behav\_2**.



```
architecture Behav_2 of FlipFlopD is
begin
```

```
end Behav_2;
```