



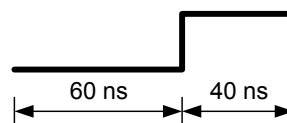
Nome: _____

Notas Importantes!

1. Verifique, para todas as questões, qual a resposta correta e assinala com um "X" a sua escolha na tabela ao lado. Por cada resposta incorreta será descontada, à cotação global, 1/3 da cotação da respetiva pergunta.
2. Pode usar até um máximo de 4 respostas duplas (por cada dupla: 0 respostas certas desconta 2/3; 1 resposta certa conta 2/3). Se usar mais de 4 duplas, serão aceites as 4 primeiras e as restantes serão consideradas respostas erradas.
3. Durante a realização do teste não é permitida a permanência junto do aluno, mesmo que desligado, de qualquer dispositivo eletrónico não expressamente autorizado (nesta lista incluem-se calculadoras, telemóveis, smartwatches e qualquer outro dispositivo de captura de imagem e/ou comunicação). A sua deteção durante a realização do exame implica a imediata anulação do mesmo.
4. Não é permitido escrever na área branca em torno da matriz de respostas.
5. Cotações: Grupo I – cada 0.5 valores; Grupo II – cada 0.75 valores

Grupo I

1. Um período de um sinal de relógio possui as especificações apresentadas na figura ao lado, pelo que a frequência e o *duty-cycle* são, respetivamente:



- ☒ a) 10 MHz e 60% ☐ b) 10 MHz e 40% ☒ c) 100 MHz e 60% ☐ d) 100 MHz e 40%

2. VHDL é uma linguagem que permite a descrição de sistemas digitais, através de

- ☒ a) processos, atribuições concorrentes, instanciação e interligação de entidades
☐ b) instanciação de entidades dentro de processos sequenciais
☐ c) atribuições combinatórias na declaração de entidades
☐ d) instanciação e interligação de componentes em *packages*

3. No *Quartus*, as etapas da compilação de um projeto incluem, por ordem,

- ☒ a) *fitter, analysis & synthesis, generate programming files*
☒ b) *fitter, generate programming files, analysis & synthesis*
☐ c) *analysis & synthesis, generate programming files, fitter*
☐ d) *analysis & synthesis, fitter, generate programming files*

4. Sendo o sinal `s_a` do tipo `std_logic_vector(1 downto 0)`, só uma das seguintes atribuições não é válida do ponto de vista sintático em VHDL. Qual?

- ☒ a) `s_a <= "T1";` ☐ b) `s_a <= "UL";` ☐ c) `s_a <= "ZZ";` ☐ d) `s_a <= "WX";`
- ↑ weak low*
↑ uninitialized
↑ impedance

5. Tendo em conta a numeração dos segmentos, apresentada na figura ao lado, qual das seguintes linhas de código deve ser usada para mostrar a letra L no display `HEX0` do kit DE2-115?

- a) `HEX0(6 downto 0) <= "0111000";` *1 → Desligado 0 → ligado*
b) `HEX0(6 downto 0) <= "1110001";`
☒ c) `HEX0(6 downto 0) <= "1000111";` *5 4 3*
d) `HEX0(6 downto 0) <= "0001110";` *1 0 0 0 1 1 1*



6. Para efeitos de simulação, a lista de sensibilidade de um processo em VHDL contém

- ☐ a) as entradas e saídas do respetivo processo
☐ b) os sinais e portos de saída do respetivo processo
☐ c) os portos de entrada nos quais são espoletados eventos através de atribuições realizadas no corpo do respetivo processo
☒ d) os sinais e portos cujos eventos desencadeiam a execução do respetivo processo

7. Analise o seguinte trecho de código VHDL quanto a erros.

```
library ieee; use ieee.std_logic_1164.all;
entity Talvez_Errada is
  port(wrong, case, when, silly : in std_logic;
        this : out std_logic_vector(3 downto 0));
end Talvez_Errada;
architecture Ri_di_cu_lous of Talvez_Errada is
begin THIS <= wrong & case & when & silly; end Ri_di_cu_lous; --??
```

Da análise do código fornecido, pode-se concluir que:

- ~~a)~~ há erros originados por uso indiferenciado de caracteres minúsculos e maiúsculos em identificadores
- b) há erros originados apenas por uso de identificadores inválidos**
- c) há erros por incompatibilidade entre tipos
- ~~d)~~ não há quaisquer erros

8. Considerando as seguintes declarações:

```
port(a : in signed(1 downto 0); ... );
...
signal b : signed(1 downto 0);  $10_2 \rightarrow 2_{10}$   $2 \times 3 = 6_{10}$   $4 \times 2^1 = 8_{10}$ 
signal c : signed(3 downto 0);  $1011_2 \rightarrow 3_{10}$   $110$ 
```

Qual das seguintes atribuições em VHDL não apresenta erros de sintaxe?

- ~~a)~~ $a \leftarrow b * c(1 \text{ downto } 0);$ $\rightarrow 2 \text{ bits}$
- ~~b)~~ $b \leftarrow a * c(1 \text{ downto } 0);$ $\rightarrow 4 \text{ bits}$
- c) $c \leftarrow (a(1) \& a(1) \& a) * (b(1) \& b(1) \& b);$
- d) $c \leftarrow a * b;$**

9. Para descrever um comparador de igualdade meramente combinatório baseado num processo, qual das seguintes alternativas é a correta?

a)	b)	c)	d)
process(a,b)	process(a,b)	process(a,b)	process(a,b)
begin	begin	begin	begin
if (a <> b) then	if (a = b) then	if (a == b) then	if (a = b) then
c <= '0';	c <= '1';	c <= '1';	c <= '1';
else	else	end if;	end if;
c <= '1';	c <= '0';	end process;	end process;
end if;	end if;		
end process;	end process;		

10. Uma testbench em VHDL *Testbench apenas simula, não interage com o exterior*

- ~~a)~~ deve ter portos de entrada e de saída
- ~~b)~~ deve ter apenas portos de entrada
- c) não deve ter nem portos de entrada nem de saída**
- d) deve ter apenas portos de saída

11. Considerando as seguintes declarações:

```
signal w, x : unsigned(7 downto 0);
signal y, z : std_logic_vector(7 downto 0);
```

Qual das seguintes atribuições em VHDL não apresenta erros de sintaxe (assumindo que no módulo onde ocorrem, apenas são usadas as packages STD_LOGIC_1164 e NUMERIC_STD)?

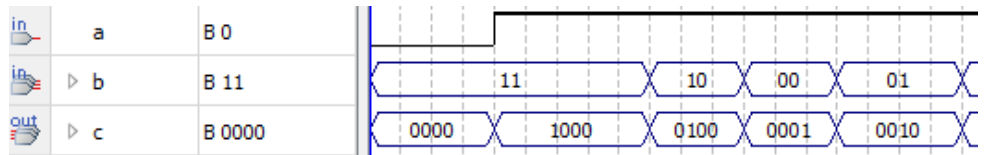
- a) $z \leftarrow \text{std_logic_vector}(w + \text{unsigned}(y));$**
- ~~b)~~ $z \leftarrow \text{std_logic_vector}(w) + y;$
- ~~c)~~ $x \leftarrow \text{unsigned}(w + y);$
- ~~d)~~ $x \leftarrow \text{unsigned}(\text{std_logic_vector}(w) + y);$

12. A gama de representação de um sinal declarado como **unsigned(7 downto 0)** é:

- a) -127 a +127
- b) -128 a +127
- c) 0 a +255**
- d) 0 a +127

Caso fosse signed seria -128, 127

13. O diagrama temporal seguinte representa os resultados de simulação de um circuito com entradas **a** e **b** e saída **c**. Este circuito é um



- ☒ a) decodificador binário 2:4 com *enable* ativo baixo
☒ b) decodificador binário 2:4 com *enable* ativo alto

- ☒ c) codificador binário 4:2 com *enable* ativo baixo
☒ d) codificador binário 4:2 com *enable* ativo alto

14. Considere o seguinte código:

```
process
begin
  if (rising_edge(clock)) then
    if (dataIn = '1') then
      dataOut <= '1';
      s_cnt <= 9;
    elsif (s_cnt /= 0) then
      s_cnt <= s_cnt - 1;
    else
      dataOut <= '0';
    end if;
  end if;
end process;
```

Para simulação correta e eficiente deste processo, a primeira linha do código (**process**) deve ser:

- ☒ a) substituída por **process (clock)**
☒ b) substituída por **process (clock, dataIn, s_cnt)**
☒ c) substituída por **process (clock, dataIn, s_cnt, dataOut)**
☒ d) mantida sem qualquer alteração

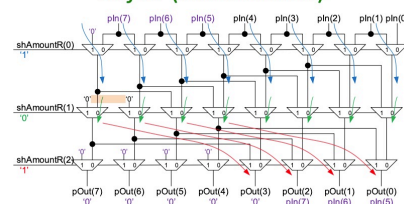
? 15. Um contador binário natural de 18 bits pode ser usado como divisor de frequência, podendo, entre outras hipóteses, dividir a frequência do *clock* de entrada por:

- ☒ a) 9 ☒ b) 18 ☒ c) 32 ☒ d) 36

16. Um *barrel shifter*

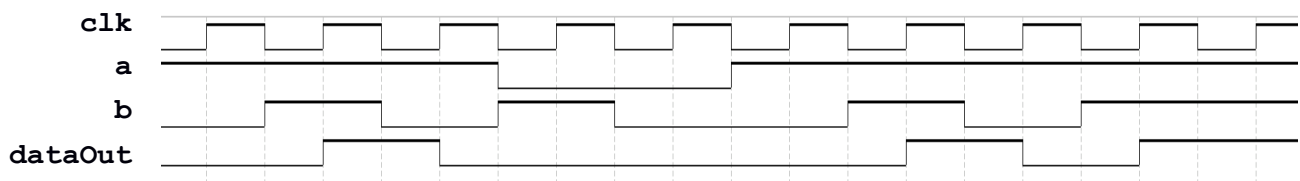
- ☒ a) nunca permite a realização de deslocamentos aritméticos
☒ b) realiza deslocamentos sem necessitar de um sinal de relógio
☒ c) é construído com *flip-flops* e lógica adicional
☒ d) necessita de “N” ciclos de relógio para deslocar “N” bits

Exemplo de Operação de um Barrel Shifter (Combinatório)



Grupo II

17. O componente cujo comportamento é apresentado no diagrama seguinte é:

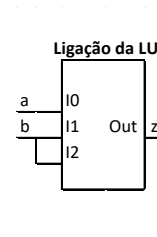


- ☒ a) uma *latch* em que **a** é a entrada de dados e **b** é o sinal de *enable*
☒ b) uma *latch* em que **b** é a entrada de dados e **a** é o sinal de *enable*
☒ c) um *flip-flop* em que **a** é a entrada de dados e **b** é o sinal de *enable*
☒ d) um *flip-flop* em que **b** é a entrada de dados e **a** é o sinal de *enable*

18. A *lookup table* (tabela de verdade) da figura ao lado implementa a seguinte função lógica:

- a) $z \leq a \text{ and } b$;
- b) $z \leq a \text{ or } b$;
- c) $z \leq a \text{ nor } b$;
- d) $z \leq a \text{ xor } b$;

Ligação da LUT



Posição (i2 i1 i0)	Out (i2 i1 i0)
0	0
1	0
2	1
3	0
4	1
5	1
6	0
7	1

19. Considere o trecho de código VHDL ao lado, relativo a um comparador com duas saídas (*gtSigned* e *gtUnsigned*). Pressupondo que *a*="1111" e *b*="0001" as saídas vão possuir, respetivamente, os valores

```
port(a, b : in std_logic_vector (3 downto 0);
...
gtSigned   <= '1' when (signed(a)   > signed(b))   else '0';
gtUnsigned <= '1' when (unsigned(a) > unsigned(b)) else '0';
```

- a) *gtSigned* <= '0' e *gtUnsigned* <= '0'
- b) *gtSigned* <= '0' e *gtUnsigned* <= '1'
- c) *gtSigned* <= '1' e *gtUnsigned* <= '0'
- d) *gtSigned* <= '1' e *gtUnsigned* <= '1'

20. Para atribuir a um sinal *dataOut* o resultado do deslocamento aritmético de 1 *bit* para a direita do sinal *dataIn*, do tipo *std_logic_vector(N-1 downto 0)*, o código VHDL a usar é:

- a) *dataOut* <= *dataIn*(*N*-2 downto 0) & *dataIn*(*N*-1);
- b) *dataOut* <= *dataIn*(*N*-2 downto 0) & *dataIn*(0);
- c) *dataOut* <= '0' & *dataIn*(*N*-1 downto 1);
- d) *dataOut* <= *dataIn*(*N*-1) & *dataIn*(*N*-1 downto 1);

21. Um codificador binário de prioridade 4:2, com sinal *validOut*, pode ser implementado com:

- a) 2 LUTs de 2 entradas
- b) 2 LUTs de 3 entradas
- c) 3 LUTs de 2 entradas
- d) 3 LUTs de 4 entradas

22. Considere o seguinte código:

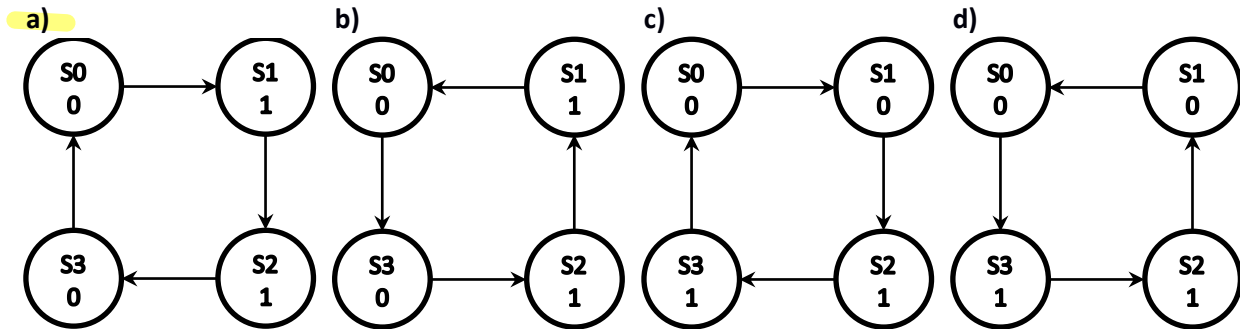
```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity customFreqDivider is
  port(clockIn : in std_logic;
        clockOut : out std_logic);
end customFreqDivider;
architecture state_machine of customFreqDivider is
  type state_t is (S0,S1,S2,S3);
  signal state : state_t := S0;
begin
  process(clockIn)
  begin
    if rising_edge(clockIn) then
      case state is
        when S0 => state <= S1; clockOut <= '0';
        when S1 => state <= S2; clockOut <= '1';
        when S2 => state <= S3; clockOut <= '1';
        when S3 => state <= S0; clockOut <= '0';
      end case;
    end if;
  end process;
end state_machine;
```

when S0
state ← S1
clockOut ← '0'

O sinal de entrada, **clockIn**, tem frequência 100 MHz e *duty-cycle* 40%. Nestas condições, a frequência e *duty-cycle* do sinal de saída, **clockOut**, são, respectivamente:

- a) 50 MHz e 40% b) 50 MHz e 50% c) 25 MHz e 50% d) 25 MHz e 40%

23. O diagrama de estados correspondente ao código da questão anterior é:



24. Uma memória RAM com a seguinte interface:

```
entity RAM is
  generic(addrBusSize : positive := 3;
    dataBusSize : positive := 8);
  port(clk      : in  std_logic;
    writeEnable : in  std_logic;
    writeAddress: in  std_logic_vector((addrBusSize - 1) downto 0);
    writeData   : in  std_logic_vector((dataBusSize - 1) downto 0);
    readAddress : in  std_logic_vector((addrBusSize - 1) downto 0);
    readData    : out std_logic_vector((dataBusSize - 1) downto 0));
end RAM;
```

foi instanciada preenchendo o **generic map** da seguinte maneira:

```
generic map(addrBusSize => 2, dataBusSize => 8)
```

No total, quantos *bits* de armazenamento contém a memória instanciada?

- a) 16 b) 32 c) 64 d) 128

25. A memória RAM da questão anterior possui a seguinte implementação:

```
architecture Behavioral of RAM is
  subtype TDataWord is std_logic_vector((dataBusSize - 1) downto 0);
  type TMemory is array (0 to (2 ** addrBusSize)-1) of TDataWord;
  signal s_memory : TMemory;
begin
  process(clk)
  begin
    if (rising_edge(clk)) then
      if (writeEnable = '1') then
        s_memory(to_integer(unsigned(writeAddress))) <= writeData;
      end if;
      readData <= s_memory(to_integer(unsigned(readAddress)));
    end if;
  end process;
end Behavioral;
```

Da análise do código fornecido podemos concluir que

- a) tanto a leitura como a escrita são síncronas c) a leitura é síncrona e a escrita é assíncrona
b) tanto a leitura como a escrita são assíncronas d) a leitura é assíncrona e a escrita é síncrona

26. O seguinte trecho de código:

```
FF: for i in 0 to 8 generate
II:   if (i rem 2 = 0) generate
E0:     entity work.flip_flop_d port map(clock=>clk, d=>dIn(i), q=>dOut(i));
      else generate
E1:     entity work.not_gate port map(s_in=>dIn(i), s_out=>dOut(i));
      end generate;
end generate;
```

instancia

- a) 9 entidades `flip_flop_d` c) 5 entidades `flip_flop_d` e 4 entidades `not_gate`
b) 9 entidades `not_gate` d) 4 entidades `flip_flop_d` e 5 entidades `not_gate`

27. A síntese dos seguintes excertos de código resulta, respetivamente, nos seguintes componentes:

i)
`o <= i0 when (s = '0') else`
`i1;`

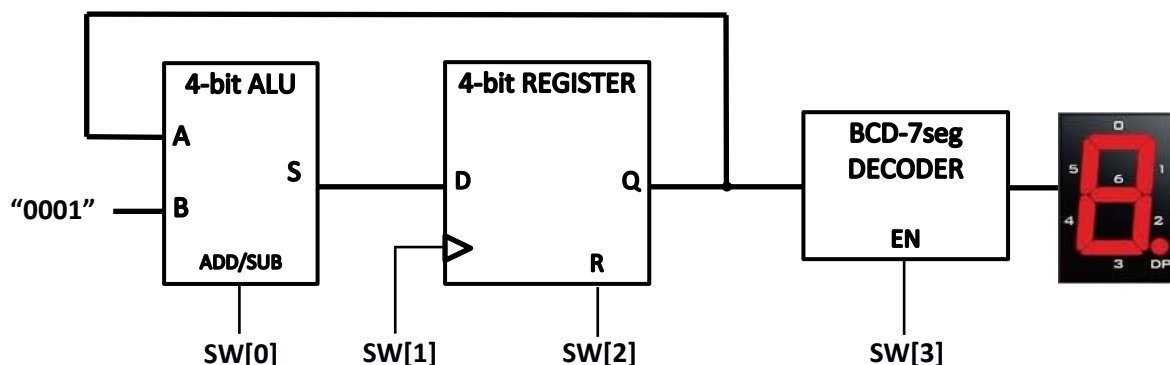
ii)
`process(c)`
`begin`
`if (rising_edge(c)) then`
`q <= q + d;`
`end if;`
`end process;`

iii)
`process(e, d)`
`begin`
`if (e = '1') then`
`q <= d;`
`end if;`
`end process;`

iv)
`process(c)`
`begin`
`if (c'event and c = '1') then`
`q <= q + 1;`
`end if;`
`end process;`

- a) multiplexador, acumulador, *latch* e contador
b) demultiplexador, somador, *flip-flop* e acumulador
c) decodificador 1:2, *flip-flop*, *latch* e *flip-flop*
d) codificador de prioridade, somador, registo e contador

28. Analise o seguinte diagrama de blocos de um sistema de contagem de pontos que permite *undo*.



Se só pudesse aplicar *debounce* a um dos quatro sinais de entrada, qual escolheria no sentido de tornar o funcionamento do sistema o mais fiável possível?

- a) SW[0] c) SW[2]
b) SW[1] d) SW[3]

29. Analise o seguinte módulo descrito em VHDL

```
entity Circuit01 is
    generic(K          : positive);
    port(clk           : in  std_logic;
          enable       : in  std_logic;
          outSecret1    : out std_logic);
end Circuit01;

architecture Behavioral of Circuit01 is
    signal s_counter : natural range 0 to K := 0;
    signal s_secret   : std_logic;
begin
    process(clk)
    begin
        if (rising_edge(clk)) then
            if (enable = '1') then
                if (s_counter >= K) then
                    s_counter <= 0;
                    s_secret  <= not s_secret;
                else
                    s_counter <= s_counter + 1;
                end if;
            end if;
        end if;
    end process;
    outSecret1 <= s_secret;
end Behavioral;
```

O trecho de código fornecido descreve um

- a) temporizador que ativa a saída **outSecret1** durante K ciclos de relógio após a detecção do nível lógico '1' na entrada **enable**
- b) temporizador que ativa a saída **outSecret1** durante 1 ciclo de relógio após a entrada **enable** permanecer ativa durante K ciclos de relógio
- c) divisor da frequência do sinal de *clock* de entrada, sendo $K+1$ o fator de divisão
- d) divisor da frequência do sinal de *clock* de entrada, sendo $2(K+1)$ o fator de divisão

30. No trecho de código

```
process(clock_50)
begin
    if rising_edge(clock_50) then
        k0 <= not key(0);
        k1 <= k0;
        s0 <= '0';
        if ((k0 = '1') and (k1 = '0')) then
            s0 <= '1';
        end if;
    end if;
end process;
s1 <= '1' when ((k0 = '1') and (k1 = '0')) else '0';
```

- a) **s0** fica a '0' quando há uma mudança qualquer no nível lógico de **key(0)**
- b) **s0** fica a '1' quando há uma mudança qualquer no nível lógico de **key(0)**
- c) sempre que **s0** passa a '1' no ciclo de relógio seguinte **s0** volta a '0'
- d) **s0** fica sempre a '0'

31. Ainda relativamente ao trecho de código da questão anterior, e desprezando os atrasos de propagação,

- a) tanto `s0` como `s1` ficam sempre a '0'
- b) `s0` e `s1` mudam de nível lógico ao mesmo tempo
- c) as mudanças de nível lógico de `s0` ocorrem um ciclo de relógio antes das de `s1`
- d) as mudanças de nível lógico de `s0` ocorrem um ciclo de relógio depois das de `s1`

32. Analise o seguinte módulo descrito em VHDL:

```
entity Circuito2 is
    generic(K          : positive := 5);
    port(clk           : in  std_logic;
          enable       : in  std_logic;
          outSecret2    : out std_logic);
end Circuito2;

architecture Behavioral of Circuito2 is
    signal s_count : natural range 0 to K := 0;
begin
    process(clk)
    begin
        if (rising_edge(clk)) then
            if (s_count = 0) then
                if (enable = '1') then
                    outSecret2 <= '1';
                    s_count    <= s_count + 1;
                else
                    outSecret2 <= '0';
                end if;
            else
                if (s_count >= K) then
                    outSecret2 <= '0';
                    s_count    <= 0;
                else
                    outSecret2 <= '1';
                    s_count    <= s_count + 1;
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```

O trecho de código fornecido descreve um

- a) temporizador que ativa a saída `outSecret2` durante K ciclos de relógio após a detecção do nível lógico '1' na entrada `enable`
- b) temporizador que ativa a saída `outSecret2` durante 1 ciclo de relógio após a entrada `enable` permanecer ativa durante K ciclos de relógio
- c) divisor da frequência do sinal de *clock* de entrada, sendo K o fator de divisão
- d) divisor da frequência do sinal de *clock* de entrada, sendo $K+1$ o fator de divisão

Questão extra:

Efetue em binário a seguinte operação: $1011_2 + 1010_2 =$ _____

Assumindo que os operandos estão representados com sinal, em complemento para 2, qual o valor dos operandos e do resultado em decimal? _____
