LSDig
16/Junho/2017
N. Mec:_____

Universidade de Aveiro – Dep. de Electrónica, Telecomunicações e Informática
Laboratório de Sistemas Digitais – Ano Letivo 2016/17 – Exame de Época Normal

Nome:_____

	a	b	c	d
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
	a	b	c	d
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
	a	b	c	d
29				
30				
31				

Notas Importantes!

1. Verifique, para todas as questões, qual a resposta correta e assinale 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 III – cada 1 valor

---

Grupo I

1. Qual o número mínimo de LUTs 3:1 necessárias para implementar um somador completo de 1 bit (*full adder*)?
a) 1 **b)** 2 **c)** 3 **d)** 4

2. Qualquer função lógica implementável numa LUT 4:1 pode também ser implementada:
a) em duas LUTs 2:1
b) em duas LUTs 3:1
c) numa LUT 5:1
d) todas as restantes respostas estão corretas

3. Se num circuito implementado na FPGA do *kit* DE2-115 a entrada correspondente ao botão KEY(0) ligar diretamente à saída LEDR(0), então:
a) LEDR(0) acende quando o botão KEY(0) não está premido
b) LEDR(0) acende quando o botão KEY(0) está premido
c) LEDR(0) não muda com o estado do botão KEY(0) porque falta um circuito de *debouncing*
d) LEDR(0) pisca com a frequência do sinal de relógio CLOCK_50 do *kit*

4. A construção **`entity`** do VHDL permite
a) descrever a implementação de um módulo
b) descrever a interface de um módulo
c) modelar a concorrência entre os vários processos definidos na sua parte declarativa
d) definir novos tipos de dados usados nos portos de um módulo

5. Em VHDL, qual das seguintes funções/macros permite converter um inteiro (**`integer`**) diretamente para **`std_logic_vector`**?
a) função **`to_integer`**
b) função **`to_unsigned`**
c) macro de conversão **`std_logic_vector`**
d) nenhuma das restantes respostas está correta

6. Em VHDL um porto corresponde:
a) à implementação dum módulo
b) a um sinal de interface dum módulo
c) a um sinal interno dum módulo
d) ao nome de um módulo numa biblioteca

7. Qual das seguintes construções de VHDL **não é** sintetizável?

a) `wait for` …
b) `if` … `then` … `else`
c) `when` … `else`
d) `case` … `is` … `when`

8. O seguinte trecho de código VHDL descreve um:

```
case inputs is     when "00"  => outputs <= "0001";
                   when "01"  => outputs <= "0010";
                   when "10"  => outputs <= "0100";
                   when others=> outputs <= "1000";
end case;
```

a) *barrel shifter* de 4 bits
b) *multiplexer* 4:2
c) codificador binário 4:2
d) descodificador binário 2:4

9. A partir do excerto de código seguinte, pode-se concluir que **`sig`** é um sinal do tipo:

```
signal s_a, s_b : signed(3 downto 0);
…
s_a <= "1010";
s_b <= "0110";
sig <= (s_a(3) & s_a) + (s_b(3) & s_b);
```

a) `signed(3 downto 0)`
b) `signed(4 downto 0)`
c) `std_logic_vector(3 downto 0)`
d) `std_logic_vector(4 downto 0)`

10. Ainda sobre o código da questão 9, o resultado da execução da seguinte expressão é:

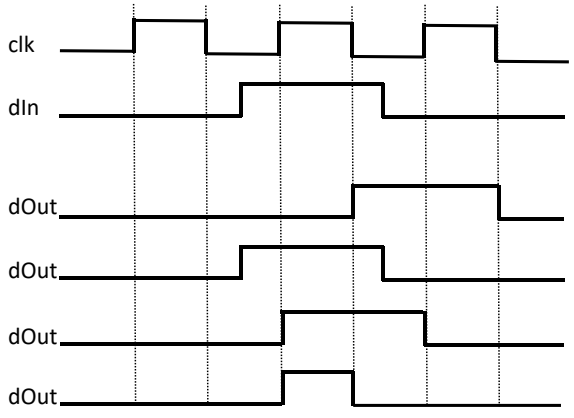
```
to_integer(s_a * s_b);
```

a) 60 **b)** -60 **c)** -36 **d)** -12

11. A linha de código VHDL `s_reg <= s_reg(N-1) & s_reg(N-1 downto 1);` descreve uma operação de:

a) deslocamento lógico à direita
b) deslocamento aritmético à esquerda
c) deslocamento aritmético à direita
d) rotação à direita

12. Qual dos diagramas temporais seguintes corresponde à simulação do código dado:




```
process (clk)
begin
   if (falling_edge(clk)) then
      dOut <= dIn;
   end if;
end process;
```

13. Considerando o seguinte excerto de código VHDL, cuja intenção é descrever um *flip-flop* tipo D, pode-se afirmar que este:

```
process(clk)
begin
  if (clk = '1') then
    dataOut <= dataIn;
  end if;
end process;
```

a) simula corretamente, sintetiza mas funciona incorretamente em hardware
b) simula, sintetiza e funciona corretamente em *hardware*
c) não simula corretamente, apesar de sintetizar e funcionar corretamente em *hardware*
d) não simula corretamente, sintetiza, e não funciona corretamente em hardware

14. Identifique a afirmação **errada** sobre um **`package`** em VHDL:
a) um **`package`** pode incluir definições de tipos
b) um **`package`** pode incluir protótipos e implementação de funções
c) um **`package`** pode fazer uso de outros **`packages`** (utilizar definições de outros **`packages`**)
d) um **`package`** pode incluir implementações (**`architectures`**)

15. Para realizar um registo de deslocamento bidirecional de 32 bits, com *reset* síncrono e possibilidade de carregamento paralelo, são necessários, no mínimo:
a) 16 *flip-flops*
b) 32 *flip-flops*
c) 64 *flip-flops*
d) 128 *flip-flops*

16. Numa memória RAM de dois portos, em que o barramento de endereços é de 4 bits e o de dados é de 16 bits, o número total de bits de armazenamento é:
a) 64 bits
b) 128 bits
c) 256 bits
d) 512 bits

Grupo II

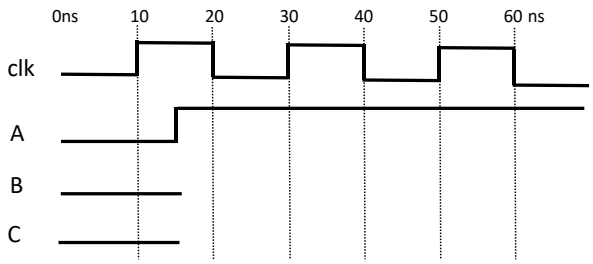
17. A análise do seguinte excerto de código VHDL permite afirmar que:

```
process(enable, counter)
begin
  if (enable = '1') then counter <= counter + 1;
  end if;
end process;
```

a) será sintetizado um contador *edge triggered* crescente
b) será sintetizado um registo de deslocamento
c) será sintetizado um acumulador *edge triggered*
d) será sintetizado um circuito cujo comportamento é imprevisível

18. Considere o seguinte excerto de código VHDL. Assumindo que **clk** e **A** evoluem de acordo com o diagrama temporal dado, o sinal **C**:

```
process (clk)
begin
   if (rising_edge(clk)) then
      B <= A;
      C <= B;
   end if;
end process;
```



a) muda de '0' para '1' aos 15 ns
b) muda de '0' para '1' aos 30 ns
c) muda de '0' para '1' aos 50 ns
d) mantém sempre o valor '0'

19. O seguinte código VHDL implementa um contador de módulo (na saída **count**):

```
entity Counter is
   port( clk, reset : in std_logic;
         count      : out std_logic_vector(3 downto 0));
end Counter;
architecture Behavioral of Counter is
   signal s_count : unsigned(3 downto 0);
   signal s_reset : std_logic;
begin
   s_reset <= reset or (s_count(3) and s_count(0));
   count <= std_logic_vector(s_count);
   process (clk)
   begin
         if (rising_edge(clk)) then
               if (s_reset = '1') then    s_count <= (others => '0');
               else                       s_count <= s_count + 1;
               end if;
         end if;
   end process;
end Behavioral;
```

a) 3 **b)** 8 **c)** 9 **d)** 10

20. No código VHDL apresentado na questão 19, o *duty cycle* da saída **count(3)** é:

a) 10% **b)** 20% **c)** 50% **d)** 80%

21. No código VHDL apresentado na questão 19, se a frequência de **clk** for 50 MHz, qual o período da saída **count(3)**?

a) 20 ns
b) 40 ns
c) 200 ns
d) é impossível calcular porque o sinal **count(3)** não é periódico

22. Considerando as seguintes atribuições concorrentes:

```
R1 <= std_logic_vector(unsigned(A) + unsigned(B));
R2 <= std_logic_vector(  signed(A) +   signed(B));
```

a) **R1** será diferente de **R2** apenas se os bits mais significativos dos operandos **A** e **B** forem opostos
b) **R1** será diferente de **R2** apenas se os bits mais significativos de ambos os operandos **A** e **B** forem '1'
c) **R1** e **R2** serão sempre diferentes
d) **R1** será sempre igual a **R2**

23. Considere o seguinte trecho de código VHDL, relativo a um comparador com duas saídas (**gtSigned** e **gtUnsigned**).

```
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';
```

Assumindo que **a="1111"** e **b="0001"** as saídas vão possuir, respetivamente, os seguintes valores

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

24. O seguinte trecho de código VHDL implementa um:

a) temporizador do tipo atraso à desoperação (saída permanece ativa durante um tempo fixo após a ativação da entrada **start**)

b) temporizador do tipo atraso à operação (saída é ativada após um tempo fixo depois da ativação da entrada **start**)

c) divisor de frequência

d) nenhuma das restantes respostas está correta

```
process(clk)
begin
  if (rising_edge(clk)) then
    if (reset = '1') then
      timerOut <= '0';
      s_count  <= 0;
    elsif (s_count = 0) then
      if (start = '1') then
        timerOut <= '1';
        s_count  <= s_count + 1;
      else
        timerOut <= '0';
      end if;
    elsif (s_count = K-1) then
      timerOut <= '0';
      s_count  <= 0;
    else
      timerOut <= '1';
      s_count  <= s_count + 1;
    end if;
  end if;
end process;
```

25. Considere uma memória RAM com a seguinte interface:

```
entity RAM is
  generic(addrBusSize : positive := 3;
          dataBusSize : positive := 4);
  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;
```

A esta memória aplica-se a afirmação seguinte:

a) é uma memória com um porto de acesso
b) é uma memória com dois portos de acesso
c) é uma memória com seis portos de acesso
d) é uma memória com **addrBusSize** portos de acesso

26. Ainda relativamente à memória da questão 25, para instanciar uma memória que armazena 64k palavras de 16 bits, a respetiva construção **`generic map`** deve ser parametrizada da seguinte forma:

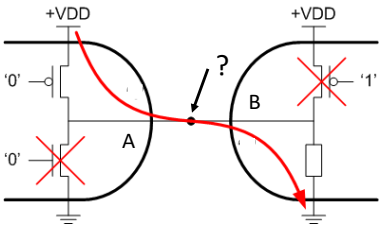
a) `addrBusSize => 64000, dataBusSize => 16`
b) `addrBusSize => 64,    dataBusSize => 64`
c) `addrBusSize => 16,    dataBusSize => 16`
d) `addrBusSize => 16,    dataBusSize => 64000`

27. Pretende-se criar uma memória ROM de tipo `TROM` que implemente um multiplicador, sem sinal, de duas quantidades de 2 bits cada. A ROM comporta-se como uma LUT em que os operandos da multiplicação são as entradas e o resultado a saída. A constante que define o conteúdo da memória pode ser inicializada de modo seguinte:

```
constant c_memory: TROM :=
```

a) `(x"0",x"1",x"2",x"3",x"4",x"5",x"6",x"7",x"7",x"6",x"5",x"4",x"3",x"2",x"1",x"0");`
b) `(x"0",x"0",x"0",x"0",x"1",x"1",x"2",x"2",x"3",x"3",x"4",x"4",x"6",x"6",x"9",x"9");`
c) `(x"0",x"0",x"0",x"0",x"0",x"1",x"2",x"3",x"0",x"2",x"4",x"6",x"0",x"3",x"6",x"9");`
d) `(x"0",x"1",x"2",x"3",x"4",x"5",x"6",x"7",x"8",x"9",x"A",x"B",x"C",x"D",x"E",x"F");`

28. Considerando o tipo de dados **`std_logic`** de VHDL, os valores "impostos" pelas portas lógicas "A" e "B", assim como o resultante no ponto "?", são respetivamente:



a) `'1'`, `'L'` e `'1'`
b) `'1'`, `'0'` e `'X'`
c) `'0'`, `'L'` e `'X'`
d) `'0'`, `'0'` e `'0'`

Grupo III

29. Considere o seguinte excerto de código VHDL:

```
ger_loop: for i in 3 downto 0 generate
     init: if (i = 3) generate
          dataOut(i) <= dataIn(i);
     else generate
          dataOut(i) <= dataIn(i+1) xor dataIn(i);
     end generate init;
end generate ger_loop;
```

Determine o valor do sinal **`dataOut(3 downto 0)`** quando **`dataIn(3 downto 0)="1100"`**.

a) `"0010"` **b)** `"0100"` **c)** `"1100"` **d)** `"1010"`

30. Considerando o seguinte código VHDL, identifique todos os módulos que constituem o *datapath* do sistema computacional **`IterCore`** (a entidade **`ShiftUnit`** implementa um registo de deslocamento).

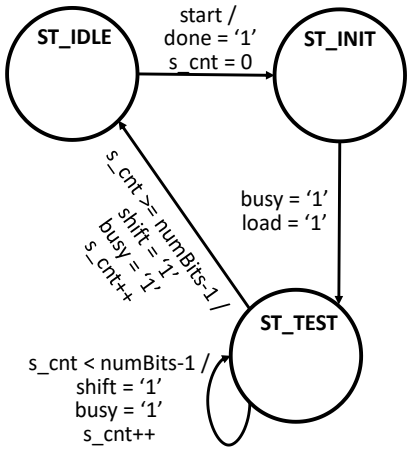
a) "**`ShiftUnit`**" e "**`Decide`**"
b) "**`ShiftUnit`**", "**`Decide`**" e "**`IterCore`**"
c) "**`decide_unit`**", "**`in_reg`**", "**`op1`**", "**`op2`**" e "**`out_reg`**"
d) "**`in_reg`**", "**`op1`**", "**`op2`**" e "**`out_reg`**"

```
entity ShiftUnit … -- registo de deslocamento à esquerda com load síncrono
--------------------------------------------------------------------------
entity Decide is
   generic(numBits    : positive := 8);
   port(clk           : in  std_logic;
        reset, start  : in  std_logic;
        busy, done    : out std_logic;
        shift, load   : out std_logic);
end Decide;
```

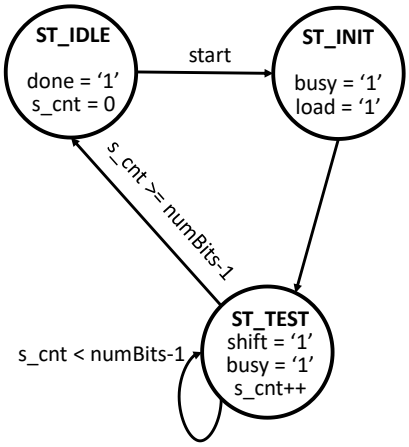
```
architecture Behavioral of Decide is
   type TState is (ST_IDLE, ST_INIT, ST_TEST);
   signal s_currentState, s_nextState : TState;
   signal s_cnt : natural range 0 to numBits;
begin
   process(clk)
   begin
      if (rising_edge(clk)) then
         if (reset = '1') then   s_currentState <= ST_IDLE;
         else                    s_currentState <= s_nextState;     end if;
     end if;
   end process;
   process(clk)
   begin
      if (rising_edge(clk)) then
         if (s_currentState = ST_IDLE) then       s_cnt <= 0;
         elsif (s_currentState = ST_TEST) then    s_cnt <= s_cnt + 1;
         end if;
      end if;
   end process;
   process(s_currentState, start, s_cnt)
   begin
      s_nextState <= s_currentState;
      case s_currentState is
      when ST_IDLE =>
         if (start = '1') then   s_nextState <= ST_INIT;    end if;
      when ST_INIT =>            s_nextState <= ST_TEST;
      when ST_TEST =>
         if (s_cnt < numBits-1) then   s_nextState <= ST_TEST;
         else                          s_nextState <= ST_IDLE;       end if;
      end case;
   end process;
   process(s_currentState)
   begin
      busy <= '0';   done <= '0';      shift <= '0';     load <= '0';
      case s_currentState is
      when ST_IDLE =>      done  <= '1';
      when ST_INIT =>      busy  <= '1';     load  <= '1';
      when ST_TEST =>      busy  <= '1';     shift <= '1';
      end case;
   end process;
end Behavioral;
--------------------------------------------------------------------------------
entity IterCore is
   generic(numBits : positive := 8);
   port(clk          : in  std_logic;
        reset, start : in  std_logic;
        busy, done   : out std_logic;
        operand      : in  std_logic_vector(numBits - 1 downto 0);
        result       : out std_logic_vector(numBits - 1 downto 0));
end IterCore;
architecture Structural of IterCore is
   signal s_xor, s_shift, s_load : std_logic;
   signal s_test, s_op : std_logic_vector(numBits downto 0);
begin
   decide_unit: entity work.Decide(Behavioral)
         generic map (numBits  => numBits)
         port map(clk => clk,    reset => reset,   start => start,
                  busy=> busy,   done  => done,    shift => s_shift,
                  load=> s_load);
```

```
    in_reg: entity work.ShiftUnit(Behavioral)
          generic map(N => numBits+1)
          port map(clk => clk,          dataIn => s_op,     sin => '0',
                   loadEn => s_load,    shiftEn => s_shift, dataOut => s_test);

    op1:  s_op  <= '0' & operand;
    op2:  s_xor <= s_test(numBits) xor s_test(numBits-1);

    out_reg: entity work.ShiftUnit(Behavioral)
          generic map(N => numBits)
          port map(clk => clk,    dataIn => (others => '0'),    sin => s_xor,
                   loadEn => s_load,  shiftEn => s_shift,  dataOut => result);
end Structural;
```

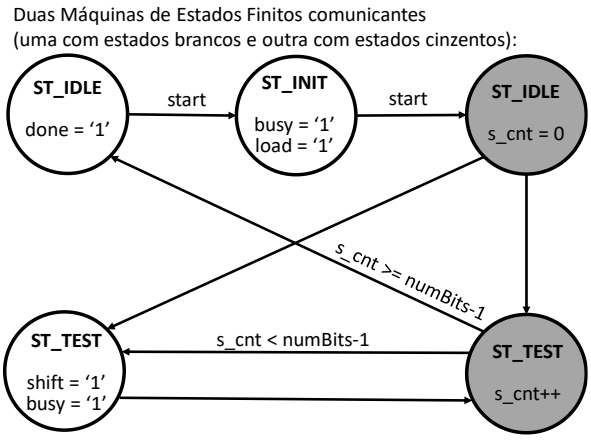
31. Qual o diagrama de estados da entidade **`Decide`** apresentada no código VHDL da questão 30? De notar que nos diagramas só são apresentadas as saídas ativas.



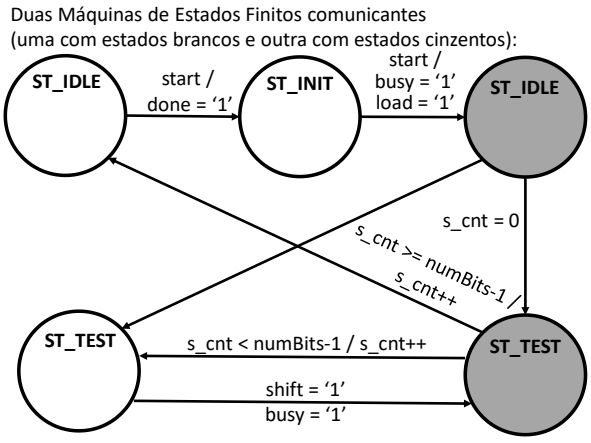
a)



b)



c)



d)

Questões extra sobre o código apresentado na questão 30:

Qual dos módulos (entidade-arquitetura) apresentados é baseado numa máquina de estados finitos? _____

Qual o tipo da máquina de estados finitos (*Mealy* ou *Moore*)? _____

Justifique: _____

_____