Laboratório de Sistemas Digitais

Trabalho Prático nº 2

Componentes combinatórios simples: descodificadores, codificadores e multiplexadores

Objetivos

- Modelação em VHDL, simulação, implementação em FPGA e teste de componentes combinatórios simples.

Sumário

Este trabalho prático está dividido em várias partes, dedicadas a diversos tipos de componentes: descodificadores, codificadores e multiplexadores. Em cada parte, pretende-se modelar, simular, implementar em FPGA e testar no kit DE2-115 o funcionamento de cada um destes componentes.

Notas importantes:

Na descrição deste trabalho prático, os passos do fluxo de projeto são descritos de forma sumária. Caso tenha dúvidas sobre algum dos passos, consulte o trabalho prático 1 para uma descrição mais completa.

No final da aula, desligue o kit e arrume-o adequadamente juntamente com os cabos e alimentador na respetiva caixa.

Parte I

1. Abra a aplicação "Altera Quartus II" e crie um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. O nome do projeto e da entidade *top-level* poderão ser ambos "Dec2_4EnDemo".

2. Crie um novo ficheiro para código fonte VHDL e introduza o seguinte código fonte.

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

entity Dec2_4En is
    port(enable  : in  std_logic;
         inputs  : in  std_logic_vector(1 downto 0);
         outputs : out std_logic_vector(3 downto 0));
end Dec2_4En;

architecture BehavEquations of Dec2_4En is
begin
    outputs(0) <= enable and (not inputs(1)) and (not inputs(0));
    outputs(1) <= enable and (not inputs(1)) and (    inputs(0));
    outputs(2) <= enable and (    inputs(1)) and (not inputs(0));
    outputs(3) <= enable and (    inputs(1)) and (    inputs(0));
end BehavEquations;
```

3. Grave o ficheiro com o nome "Dec2_4En.vhd".

4. Identifique o componente modelado e construa no seu *log book* a respetiva tabela de verdade.

5. Efetue a simulação do componente modelado, realizando para tal os seguintes passos:
- selecione o ficheiro "Dec2_4En.vhd" como o *top level* do projeto, de forma a efetuar a simulação apenas deste módulo.
- execute a opção "*Analysis & Synthesis*" para, entre outros aspetos, verificação da correção sintática e estrutura do projeto.
- crie um ficheiro VWF de suporte à simulação.
- selecione os portos a usar na simulação e especifique os vetores de entrada a usar (sugestão: utilize a opção "*Random Values*" para estabelecer os valores da entrada "**inputs**" e o método que usou na aula 1 para definir o valor da entrada "**enable**" ao longo do tempo).
- grave o ficheiro com o nome "Dec2_4En.vwf", execute a simulação e analise os resultados.

6. Crie um novo ficheiro VHDL, chamado "Dec2_4EnDemo.vhd", onde deverá instanciar o componente modelado **Dec2_4En** e associar os respetivos portos a pinos concretos da FPGA do kit de desenvolvimento que vai usar para o testar (entradas ligadas a interruptores e saídas ligadas a LEDs do kit DE2-115). Por conveniência é fornecido o seguinte código base para este efeito, mas que possui intencionalmente alguns erros de sintaxe. Edite e corrija os erros de sintaxe e no final grave o ficheiro.

```
entity Dec2_4EnDemo is
    port(SW   : std_logic_vector(2 downto 0);
         LEDG : std_logic_vector(3 downto 0));

architecture Shell of Dec2_4EnDemo is
begin
    system_core : work.Dec2_4En(BehavEquations)
                  port map(enable  <= SW(2);
                           inputs  <= SW;
                           outputs => LEDG(3 downto 0);
end Dec2_4EnDemo;
```

7. Selecione o ficheiro "Dec2_4EnDemo.vhd" como o novo *top level* do projeto.

8. Importe as definições de pinos da FPGA do kit DE2-115 (ficheiro "DE2_115.qsf").

9. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

10. No final do processo de compilação, programe a FPGA e teste o funcionamento do componente.

11. Visualize e interprete o diagrama esquemático do circuito resultante da síntese lógica, usando para tal a ferramenta disponível no menu "*Tools->Netlist Viewers->RTL Viewer*".

12. Visualize agora e interprete o diagrama esquemático do circuito resultante do mapeamento da *netlist* nas primitivas da FPGA, usando para tal a ferramenta disponível no menu "*Tools->Netlist Viewers->Technology Map Viewer (Post-Mapping)*". Efetue uma análise comparativa deste diagrama com o visualizado no ponto anterior.

13. Edite o ficheiro "Dec2_4En.vhd" acrescentando no final a seguinte arquitetura (implementação) alternativa ("**BehavAssign**"). Nesta nova arquitetura são usadas atribuições condicionais para modelar o componente em vez de atribuições convencionais (concorrentes e incondicionais) usadas no caso anterior para escrever as equações lógicas das saídas.

```
architecture BehavAssign of Dec2_4En is
begin
    outputs <= "0000" when (enable = '0')  else
               "0001" when (inputs = "00") else
               "0010" when (inputs = "01") else
               "0100" when (inputs = "10") else
               "1000";
end BehavAssign;
```

14. No ficheiro "Dec2_4EnDemo.vhd" altere a instanciação do componente de forma a que na instanciação seja agora usada a arquitetura "**BehavAssign**" em vez da "**BehavEquations**". Repita os pontos 9 a 13 realizando a síntese, implementação, programação da FPGA e teste para a implementação alternativa (arquitetura "**BehavAssign**") do componente.

15. Feche a aplicação de programação da FPGA e seguidamente o projeto.

[TPC] Repita os pontos 9 a 13 para a seguinte arquitetura "**BehavProc**" (baseada num processo em VHDL). Efetue a simulação e compilação prévia do projeto no seu PC antes de se dirigir à sala do DETI onde se encontram os kits.

```
architecture BehavProc of Dec2_4En is
begin
    process(enable, inputs)
    begin
        if (enable = '0') then
            outputs <= "0000";
        else
            if (inputs = "00") then
                outputs <= "0001";
            elsif (inputs = "01") then
                outputs <= "0010";
            elsif (inputs = "10") then
                outputs <= "0100";
            else
                outputs <= "1000";
            end if;
        end if;
    end process;
end BehavProc;
```

Parte II

1. Escreva no seu *log book* uma entidade e uma arquitetura em VHDL para um multiplexador de 2->1, cuja interface deve ser a representada na Figura 1 e a implementação deve ser baseada num processo em VHDL (*process*).

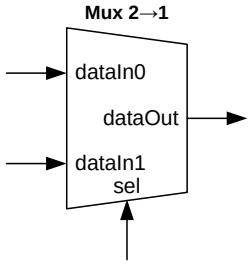


Figura 1 – Interface do multiplexador 2->1.

2. Crie no "Altera Quartus II" um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. O nome do projeto e da entidade *top-level* poderão ser ambos "Mux2_1Demo".

3. Crie um novo ficheiro para código fonte VHDL, introduza o código fonte do multiplexador do ponto 1 e grave-o com o nome "Mux2_1.vhd".

4. Efetue a simulação do multiplexador realizando todos os passos necessários e especificando uma sequência de vetores de entrada que permita validar adequadamente o seu comportamento.

5. Crie um novo ficheiro VHDL, chamado "Mux2_1Demo.vhd", que irá servir para instanciar o multiplexador e associar os respetivos portos a pinos concretos da FPGA do kit de desenvolvimento que vai usar para o testar (sugere-se que ligue as entradas de dados a interruptores, a entrada de seleção a um botão e as saídas a LEDs do kit DE2-115).

6. Importe as definições de pinos da FPGA do kit de desenvolvimento (ficheiro "DE2_115.qsf").

7. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

8. No final do processo de compilação, programe a FPGA e teste o funcionamento do multiplexador.

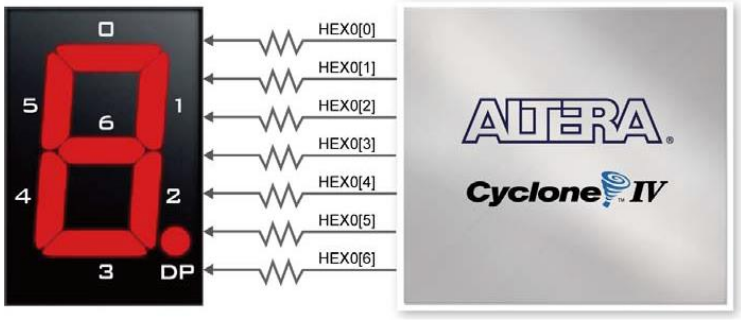
9. Feche a aplicação de programação da FPGA e seguidamente o projeto.

10. Repita a parte II para um multiplexador de 4->1, em que a entrada de seleção **`sel`** deve ser um vetor de 2 bits.

[TPC] Repita a parte II, mas usando agora uma abordagem de modelação do multiplexador de 4->1 baseada em atribuições condicionais. Efetue a simulação e compilação prévia do projeto no seu PC antes de se dirigir à sala do DETI onde se encontram os kits.

Parte III

Um módulo muito útil em sistemas digitais é um descodificador binário -> 7 segmentos, o qual converte uma palavra binária de entrada de 4 bits, representando um valor entre 0 e 9 (ou entre 0 e 15), num padrão de saída de 7 bits destinado a controlar os vários LEDs de um *display* de 7 segmentos. Nesta parte do trabalho prático pretende-se implementar um sistema de visualização simples para um *display* de 7 segmentos em que a entrada do descodificador é controlada por interruptores do kit DE2-115 e as saídas serão ligadas ao *display* HEX0. Considere o esquema de ligações entre este *display* da placa DE2-115 e a FPGA mostrado na figura seguinte. Os restantes *displays* do kit possuem esquemas de ligação semelhantes – ver manual do utilizador disponível no site de LSDig.



Uma possível implementação em VHDL de um descodificador binário para 7 segmentos é a seguinte (usando a atribuição seletiva baseada na construção **`with...select`** de VHDL).

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

entity Bin7SegDecoder is
    port(binInput : in  std_logic_vector(3 downto 0);
         decOut_n : out std_logic_vector(6 downto 0));
end Bin7SegDecoder;

architecture Behavioral of Bin7SegDecoder is
begin
    with binInput select
        decOut_n <= "1111001" when "0001",   --1
                    -- determine o valor das saídas para o dígito 2
                    "0110000" when "0011",   --3
                    "0011001" when "0100",   --4
                    "0010010" when "0101",   --5
                    "0000010" when "0110",   --6
                    "1111000" when "0111",   --7
                    "0000000" when "1000",   --8
                    "0010000" when "1001",   --9
                    "0001000" when "1010",   --A
                    "0000011" when "1011",   --B
                    "1000110" when "1100",   --C
                    "0100001" when "1101",   --D
                    "0000110" when "1110",   --E
                    "0001110" when "1111",   --F
                    "1000000" when others;   --0
end Behavioral;
```

1. Analise o código VHDL do descodificador binário -> 7 segmentos fornecido e identifique o nível lógico que é necessário aplicar (do lado da FPGA) para ativar um segmento.

2. Note que a descodificação é apresentada para todos os dígitos hexadecimais, exceto para o dígito "2". Determine o valor das saídas para esse dígito e efetue as modificações necessárias na linha de código comentada.

3. Crie no "Altera Quartus II" um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. O nome do projeto e da entidade *top-level* poderão ser ambos "DisplayDemo".

4. Crie um novo ficheiro para código fonte VHDL, introduza o código fonte do descodificador **`Bin7SegDecoder`** e grave-o com o nome "Bin7SegDecoder.vhd".

5. Crie um novo ficheiro VHDL, chamado "DisplayDemo.vhd", para instanciar o descodificador **`Bin7SegDecoder`** e associar os respetivos portos a pinos concretos da FPGA do kit de desenvolvimento que vai usar para o testar. Ligue as entradas a interruptores do kit e as saídas aos sinais de controlo dos segmentos do *display* HEX0.

6. Importe as definições de pinos da FPGA do kit de desenvolvimento (ficheiro "DE2_115.qsf").

7. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

8. No final do processo de compilação, programe a FPGA e teste o funcionamento do sistema.

9. Introduza no projeto as alterações necessárias para que as entradas do descodificador **`Bin7SegDecoder`** também sejam visualizadas em quatro LEDs do kit (sugestão: os portos de entrada ligados aos interruptores passam também a ligar a portos de saída associados aos LEDs).

10. Importe novamente o ficheiro "DE2_115.qsf" e repita a síntese e implementação do projeto através do comando "*Compile Design*". No final do processo de compilação programe a FPGA e teste o funcionamento do sistema.

11. Adicione ao descodificador uma entrada de **`enable`** que, quando ativa habilite o funcionamento normal do descodificador e, quando inativa iniba (desative) todas as saídas, apagando todos os segmentos, independentemente do valor de entrada.

11. Altere o *top-level* do projeto de forma a que a entrada **`enable`** do descodificador seja controlada por um interruptor do kit.

12. Volte a importar o ficheiro "DE2_115.qsf", a sintetizar e a implementar o projeto através do comando "*Compile Design*". No final programe a FPGA e teste o funcionamento do sistema.

13. Feche a aplicação de programação da FPGA e seguidamente o projeto.

Parte IV

1. Escreva no seu *log book* a tabela de verdade de um codificador de prioridade 4->2 com indicação de saída válida e cuja interface é representada na Figura 2.

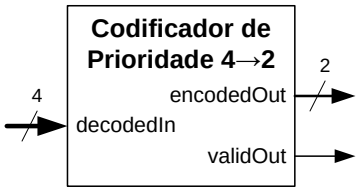


Figura 2 – Interface do codificador de prioridade.

2. Escreva no seu *log book* uma entidade e uma arquitetura em VHDL para o codificador de prioridade, cuja implementação deve ser baseada num processo (*process*).

3. Crie no "Altera Quartus II" um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. O nome do projeto e da entidade *top-level* poderão ser ambos "PEnc4_2Demo".

4. Crie um novo ficheiro para código fonte VHDL, introduza o código fonte do codificador do ponto 2 e grave-o com o nome "PEnc4_2.vhd".

5. Efetue a simulação do codificador realizando todos os passos necessários e especificando uma sequência de vetores de entrada que permita validar adequadamente o seu comportamento.

6. Crie um novo ficheiro VHDL, chamado "PEnc4_2Demo.vhd", que irá servir para instanciar o codificador e associar os respetivos portos a pinos concretos da FPGA do kit de desenvolvimento que vai usar para o testar (sugere-se que ligue as entradas a interruptores e as saídas a LEDs do kit DE2-115).

7. Importe as definições de pinos da FPGA da kit DE2-115 (ficheiro "DE2_115.qsf").

8. Efetue a síntese e implementação do projeto através do comando "*Compile Design*".

9. No final do processo de compilação, programe a FPGA e teste o funcionamento do codificador.

10. Feche a aplicação de programação da FPGA e seguidamente o projeto.

PDF criado em 18/02/2015 às 23:20:50