Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico
Universidade de Lisboa

Projecto de Sistemas Digitais

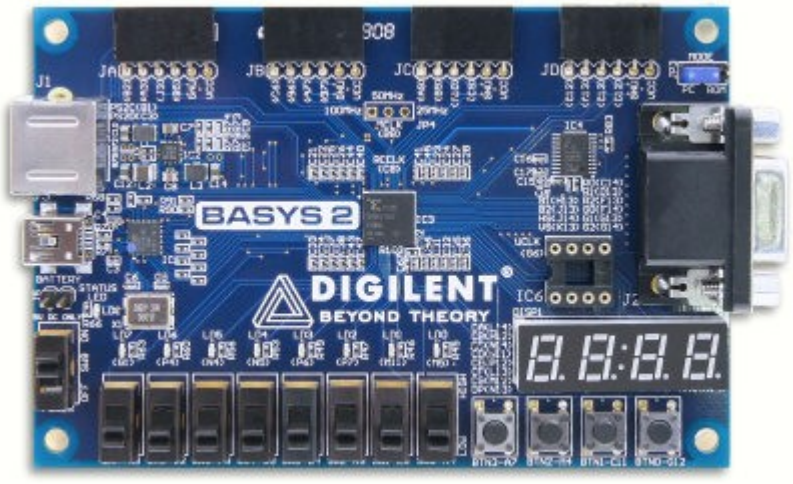
Guia de Introdução ao Laboratório

Paulo Flores, Horácio Neto
Setembro de 2011

1. Introdução

Pretende-se com este guia introduzir as ferramentas de Projecto da Xilinx e as placas de desenvolvimento que serão usadas nos laboratórios, através da realização de um circuito exemplo.

Neste laboratório serão utilizadas as placas de desenvolvimento Basys2 da Digilent (http://www.digilentinc.com/ http://aliatron.pt/e-biz/product_info.php/cPath/47_191/products_id/1196).



Placa de Desenvolvimento Basys2.

O circuito exemplo realiza uma de 3 operações (ADD, SUB, AND), entre o operando de entrada e o valor existente no acumulador. A operação a realizar é seleccionada pela activação de um botão de pressão.

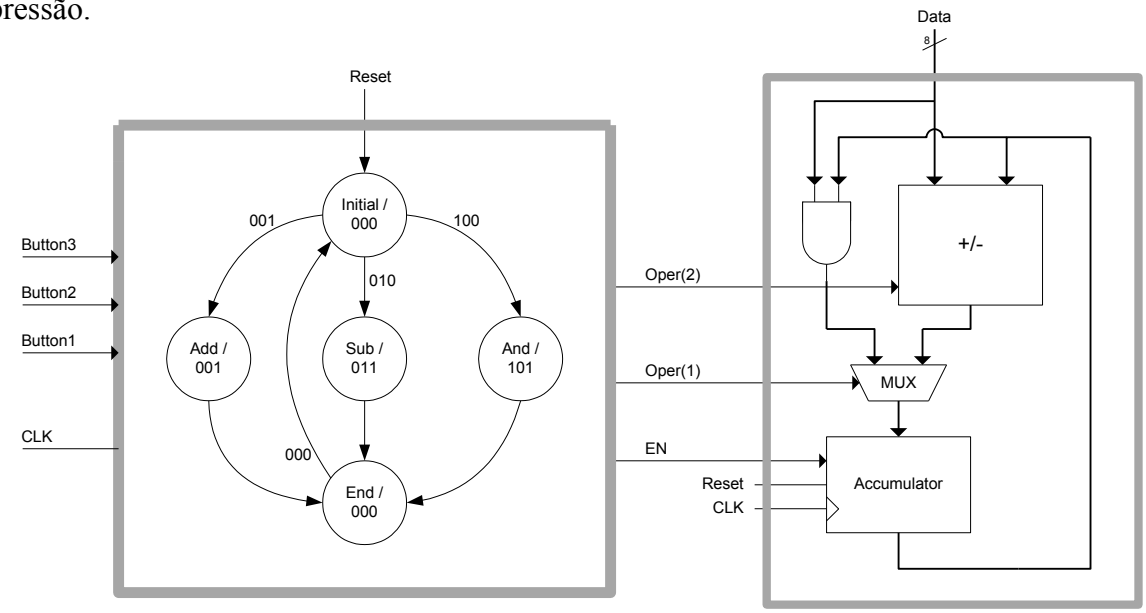


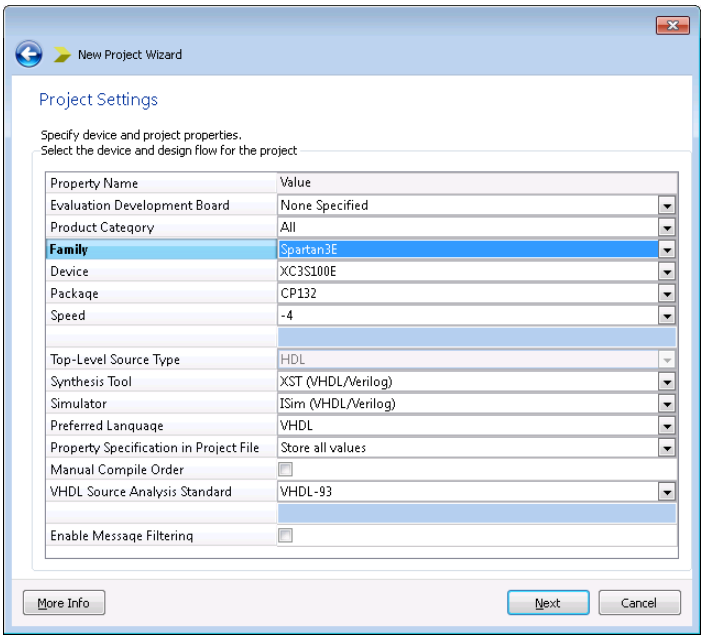
Figura 1. Unidades de Dados e de Controlo do circuito exemplo.

2. Criação do Projecto

O sistema de Projecto utilizado para especificar, simular e sintetizar o circuito é o *ISE - Project Navigator da Xilinx*. Para iniciar este programa clique no ícone Xilinx ISE.

Para criar um novo projecto seleccione o menu *File → New Project*

Na janela que aparece escolha uma localização para o projecto (*Project Location*) numa directoria do disco local (p.ex. C:\TEMP) e depois o nome que quer dar ao projecto (p.ex. proj1). Verifique que o *Top-Level Source Type* seleccionado é HDL, que a *Synthesis Tool* é o XST e que o *Simulator* é o ISim.
Seleccione *Next* > para passar ao passo seguinte.



Certifique-se que a propriedades do dispositivo têm os valores apresentados na tabela ao lado. (As placas Basys2 têm um dispositivo Spartan3E XC3S100E-4-CP132; as placas Basys têm um dispositivo Spartan3E XC3S100E-4-TQ144). Escolha correctamente o dispositivo a usar. Pode alterá-lo posteriormente nas Propriedades do Projecto.

Seleccione *Next* em todos os passos seguintes, sem introduzir nenhum valor, e termine seleccionando *Finish*. Os componentes do projecto podem ser criados/adicionados posteriormente.

3. Criação de Novos Circuitos no Projecto.

Para criar um novo componente do projecto seleccione a opção *Project → New Source*

No 1º passo (*New Source Wizard → Select Source Type*), da janela que se abre, escolha o tipo de ficheiro como *VHDL Module*, para a descrição do circuito em VHDL (ou *Schematic* para descrição do circuito através de esquemas). Note que num dado projecto podem coexistir ficheiros de vários tipos. Introduza o nome do novo ficheiro a criar, que deve ser igual ao nome da entidade VHDL que vai ser descrita. Seleccione *Next* para passar ao passo seguinte.

No 2º passo (*New Source Wizard → Define Module*) verifique se os nomes da entidade (*Entity Name*) e da arquitectura (*Architecture Name*) do componente que vai especificar em VHDL estão correctos. Introduza o nome dos portos de entrada e saída do circuito, e as respectivas direcções (veja figura abaixo, para criar o componente *datapath*). Para os portos que são barramentos (*buses*) defina também qual a posição dos bits mais e menos significativo.

Note que estes dados apenas vão ser utilizados no *template* de VHDL que será apresentado quando da criação da especificação inicial do circuito, podendo depois ser alterados por edição de texto.

Seleccione *Next* para passar ao passo seguinte.

No 3º passo (*New Source Wizard - Summary*) certifique-se que todos os dados introduzidos estão correctos e seleccione *Finish* para terminar a criação do novo ficheiro VHDL.

A janela com a hierarquia de projecto (*sources for synthesis*) permite de forma simples verificar a hierarquia do projecto e aceder directamente à edição das especificações respectivas. Pode utilizar directamente o editor de texto interno, ou utilizar qualquer outro editor de texto para (re)escrever, alterar e/ou completar a especificação do componente.

Seguindo os passos acima indicados crie um componente *datapath.vhd* com a especificação da figura 2.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity datapath is
  port ( a    : in std_logic_vector (7 downto 0);
         oper : in std_logic_vector (1 downto 0);
         clk, en_accum, rst_accum : in std_logic;
         res : out std_logic_vector (7 downto 0));
end datapath;

architecture behavioral of datapath is
  signal addsub, log_and, res_alu : std_logic_vector (7 downto 0);
  signal accum : std_logic_vector (7 downto 0) := (others => '0');
begin

  -- adder/subtracter
  addsub <= accum + a when oper(0)='0' else
            accum - a;

  -- logic unit
  log_and <= a and accum;

  -- multiplexer
  res_alu <= addsub when oper(1)='0' else
             log_and ;

  -- accumulator
  process (clk)
  begin
    if clk'event and clk='1' then
      if rst_accum='1' then
        accum <= X"00";
      elsif en_accum = '1' then
        accum <= res_alu;
      end if;
    end if;
  end process;

  -- saida
  res <= accum;
end behavioral;
```

Figura 2. Especificação da Unidade de Dados

4. Adição de componentes pré-especificados

Para adicionar ao projecto um componente já especificado seleccione a opção *Project → Add Source* (se o ficheiro já estiver na directoria de projecto) ou a opção *Project → Add Copy of Source* (para criar uma cópia, na directoria de projecto, do ficheiro original).

Seguindo os passos acima indicados adicione ao projecto o ficheiro, fornecido na página da cadeira, *control.vhd* (contém a descrição da máquina de estados que controla o funcionamento da unidade de dados especificada em 3).

5. Instanciação de componentes.

Adicione ao projecto o ficheiro *circuito.vhd*. Esta especificação junta as unidades de dados e de controlo, instanciando os componentes *datapath* e *control*, do nível hierárquico inferior.

Habitualmente é necessário incluir as definições do(s) componente(s) e da sua instanciação. As *templates* respectivas podem ser geradas automaticamente seleccionando, por exemplo, o componente *datapath.vhd* e, na janela *Processes*, clicando em *Design Utilities → View HDL Instantiation Template*. (Não se esqueça de verificar em *Properties* que a linguagem escolhida é VHDL e não *Verilog*)

As *templates* geradas automaticamente com a definição do componente e com a sua instanciação, podem ser incluídas directamente (*cut-and-paste*) na especificação do componente hierarquicamente acima:

```
-- VHDL Instantiation Created from source file datapath.vhd -- 19:57:25
02/23/2010
--
-- Notes:
-- 1) This instantiation template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the instantiated module
-- 2) To use this template to instantiate this entity, cut-and-paste and then
edit

COMPONENT datapath
  PORT(
    a : IN std_logic_vector(7 downto 0);
    oper : IN std_logic_vector(1 downto 0);
    clk : IN std_logic;
    en_accum : IN std_logic;
    rst_accum : IN std_logic;
    res : OUT std_logic_vector(7 downto 0)
    );
END COMPONENT;

Inst_datapath: datapath PORT MAP(
  a => ,
  oper => ,
  clk => ,
  en_accum => ,
  rst_accum => ,
  res =>
);
```

6. Síntese do Circuito.

A janela do lado esquerdo (*Processes*) apresenta os processos e operações que podem ser aplicadas ao circuito seleccionado. Para sintetizar o circuito faça um duplo clique em *Synthesize XST*. Se não ocorrerem erros durante o processo de síntese um sinal de executado aparece a verde. Se existirem erros ou avisos aparece uma cruz a vermelho ou uma exclamação a amarelo. Neste caso deve ser verificado o relatório do processo de síntese, na janela inferior, onde estão indicados os erros com uma marca vermelha e os avisos com uma marca amarela. Se clicar no *link* com a indicação do erro este é localizado na respectiva especificação VHDL.

7. Simulação Comportamental do Circuito.

A simulação do circuito só deve ser feita depois de se garantir que a respectiva descrição VHDL não apresenta erros e pode ser sintetizável pela ferramenta de síntese (*Synthesize XST→ Check Syntax*).

Para realizar a simulação do circuito é conveniente criar uma bancada de teste (*testbench*) especificada também em VHDL. Esta é a forma mais simples e normalizada de descrever os estímulos de simulação e verificar o comportamento do circuito, pois utiliza a própria linguagem VHDL. Assim, deve ser criada uma entidade de teste cuja arquitectura instancia o componente de topo do circuito a testar e gera os sinais necessários para a sua verificação.

Para adicionar uma bancada de teste ao projecto seleccione *Project → New Source...*

No 1º passo (*New Source Wizard → Select Source Type*), da janela que se abre, escolha o tipo de ficheiro como *VHDL TestBench*. Introduza o nome da bancada de teste a criar, em geral deverá será igual ao nome do circuito seguido de tb, p.ex. *circuito_tb*. Seleccione *Next* para passar ao passo seguinte.

No 2º passo (*New Source Wizard → Associate Source*) seleccione o nome do circuito (entidade) para o qual se vai criar a bancada de teste. Seleccione *Next* para passar ao passo seguinte.

No 3º passo (*New Source Wizard → Summary*) certifique-se que todos os dados introduzidos estão correctos e seleccione *Finish* para terminar a criação da nova bancada de teste.

A especificação da bancada de teste deve ser completada com as sequências de teste necessárias para verificar o funcionamento correcto do circuito.

Siga os passos indicados para gerar o ficheiro de teste especificado na figura 3 da página seguinte, associando-o ao componente especificado em *circuito.vhd*. Necessita apenas de incluir as variações nas entradas (linhas a **bold**). Note que as variações das entradas externas introduzidas manualmente devem respeitar os tempos de *setup/hold* nos registos, i.e. não podem coincidir com as transições (positivas) do relógio.

Na janela da hierarquia de projecto escolha *Behavioral Simulation* e seleccione o ficheiro com a bancada de teste. Na janela de *Processes* escolha *Simulate Behavioral Model*. Abrir-se-á uma janela com os resultados da simulação, onde poderá visualizar as formas de onda dos sinais de entrada/saída do componente de topo (figura 4). Note que a simulação comportamental não considera os atrasos do circuito. Deste modo, todas as variações nas saídas aparecem imediatamente após as variações de relógio. Assim, a simulação comportamental apenas permite uma verificação simples da funcionalidade do circuito, não permitindo detectar erros resultantes de atrasos no circuito, nem probelmas de sincronização temporal.

Poderá visualizar outros sinais seleccionando-os na janela *Sim Hierarchy*, arrastando-os para a janela de simulação e repetindo-a (*Restart* seguido de *Run for Specified Time*).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY circuito_tb IS
END circuito_tb;

ARCHITECTURE behavior OF circuito_tb IS
  -- Component Declaration for the Unit Under Test (UUT)
  COMPONENT circuito
  PORT(
      clk : IN  std_logic;
      rst : IN  std_logic;
      instr : IN  std_logic_vector(2 downto 0);
      data_in : IN  std_logic_vector(7 downto 0);
      res : OUT  std_logic_vector(7 downto 0)
      );
  END COMPONENT;

  --Inputs
  signal clk : std_logic := '0';
  signal rst : std_logic := '0';
  signal instr : std_logic_vector(2 downto 0) := (others => '0');
  signal data_in : std_logic_vector(7 downto 0) := (others => '0');
  --Outputs
  signal res : std_logic_vector(7 downto 0);

  -- Clock period definitions
  constant clk_period : time := 10 ns;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: circuito PORT MAP (
          clk => clk,
          rst => rst,
          instr => instr,
          data_in => data_in,
          res => res
        );

   -- Clock process definitions
   clk_process: process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for clk_period*10;
      -- insert stimulus here
      rst <= '1' after 20 ns,
             '0' after 40 ns;
      data_in <= X"67" after  40 ns,
                 X"12" after 120 ns,
                 X"C3" after 200 ns;
      instr <= "001" after  40 ns,
               "000" after  80 ns,
               "010" after 120 ns,
               "000" after 160 ns,
               "100" after 200 ns,
               "000" after 300 ns;
      wait;
   end process;
END;
```

Figura 3. Especificação das Entradas de Teste

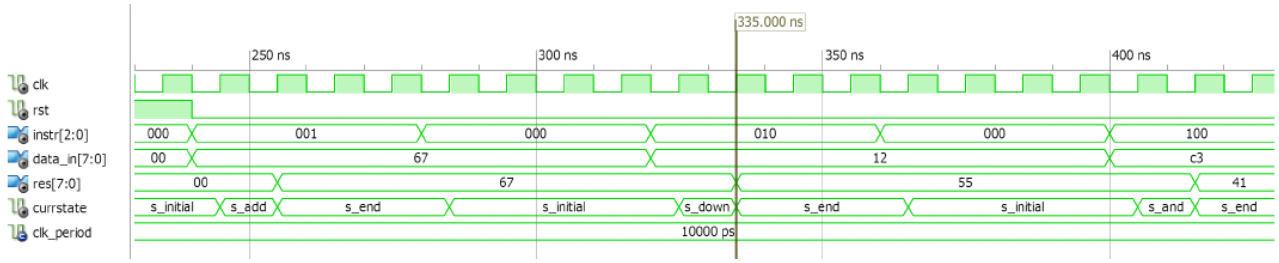


Figura 4. Simulação Comportamental do Circuito

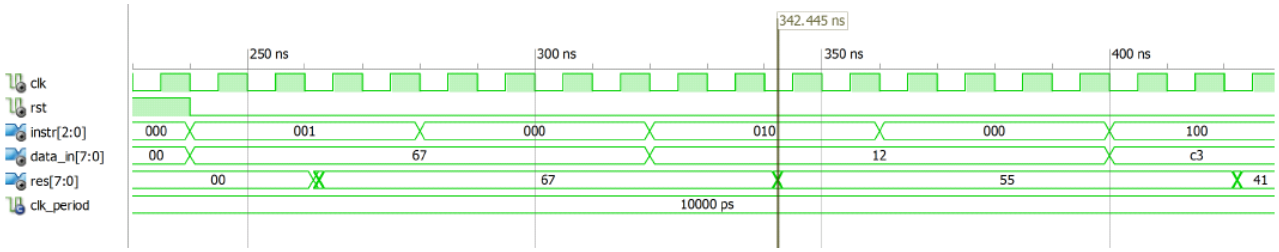


Figura 5. Simulação Temporal do Circuito

8. Simulação Temporal do Circuito

A simulação temporal do circuito tem como objectivo simular o circuito tendo em conta todos os atrasos existentes em cada um dos sinais do circuito. Estes atrasos resultam do mapeamento do circuito em portas lógicas com atrasos específicos, e também dos atrasos introduzidos pelas interligações dessas portas (que podem ser muito significativos no caso das FPGA). Só assim é possível verificar que os atrasos introduzidos pela implementação do circuito numa dada tecnologia não vão alterar a sua funcionalidade.

Para esta simulação pode, e deve, ser usada a mesma bancada de teste utilizada na simulação comportamental. Para tal deve seguir os passos indicados para a simulação comportamental, mas escolhendo *Post-Route Simulation.* Neste caso obterá os resultados indicados na figura 5. Note que os sinais de saída variam com algum atraso após a variação de relógio.

Após o mapeamento físico do circuito, os sinais internos podem passar a ter designações irreconhecíveis. Para manter as designações dos portos dos componentes de mais baixo nível deve escolher *Keep Hierarchy* nas opções de Síntese e de Mapeamento.

9. Realização do circuito a concretizar na placa de desenvolvimento

Para testar o circuito na placa tem de juntar ao projecto os componentes que permitem concretizar a interface à placa:

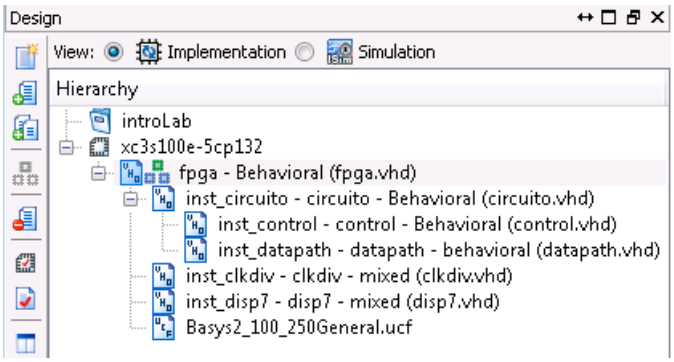
fpga.vhd - componente de topo do circuito exemplo, com as interfaces à placa,

clkdiv.vhd - componente que faz a divisão de frequência para gerar os relógios internos

disp7.vhd - interface aos *displays* de 7 segmentos

Basys2_100_250General.ucf - atribuição de pinos aos sinais de entrada/saída do chip.
(ou *BasysRevEGeneral.ucf*, se usar a placa Basys)

Na janela da hierarquia de projecto escolha *Synthesis/Implementation* e seleccione o ficheiro com o componente de topo.



Confirme que todos os componentes constam da janela de Projecto e que a hierarquia do circuito está correctamente estabelecida.

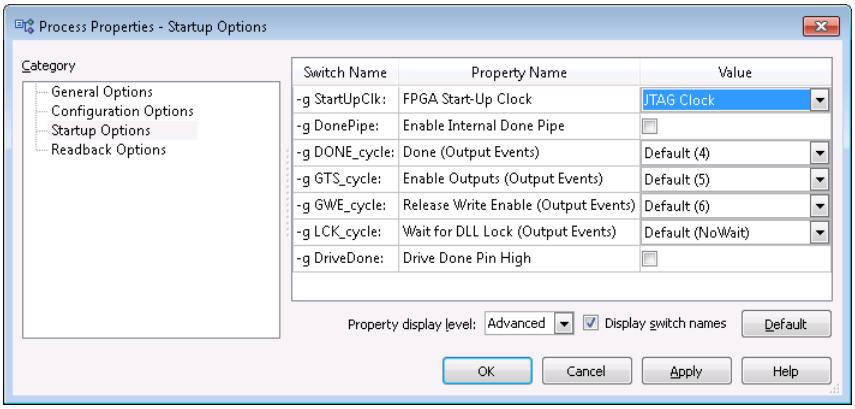
A inclusão do ficheiro .ucf correcto é **obrigatória** e deve ser sempre verificada.

Faça a síntese do circuito, clicando em *Synthesize – XST*.

10. Implementação do Projecto

Na janela de *Processes* escolha *Implement Design* seguido de *Generate Programming File*.

Confirme que nas *Startup Options* (em *Process Properties*) está escolhido *JTAG clock*.



Se não existirem problemas de implementação, é gerado o ficheiro *fpga.bit*, e o circuito está pronto a ser carregado na placa.

11. Configuração da FPGA com o Circuito

Para configuração do dispositivo na placa de desenvolvimento utilize o programa Adept da Digilent. Escolha o ficheiro *fpga.bit* criado anteriormente na janela *Config* e clique em *Program*.

O programa vai agora carregar a configuração do circuito para o dispositivo. Após este carregamento, o circuito está implementado no dispositivo. Pode verificar e demonstrar o seu funcionamento.

12. Referências

Listam-se em seguida algumas formas de obter mais informação sobre síntese de circuitos usando este ambiente de projecto.

- Consultar o manual da ferramenta de síntese *Xilinx Synthesis Technology - XST User Guide* que se encontra disponível (em formato pdf) em http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/xst.pdf , nomeadamente os capítulos 3 - *XST HDL Coding Techniques* e 14 - *XST VHDL Language Support* .

- Consultar e utilizar os *Language Templates* que se encontram no ícone "lâmpada" da barra de ferramentas do Xilinx ISE, em particular os *templates* de VHDL para síntese.

- Ler a documentação existente no menu de Help da ferramenta, em particular em *Help → Software Manuals* (em formato pdf).