Circuitos Digitais em VHDL

Versão 2022

INTRODUÇÃO

Nesta experiência é dada uma introdução da linguagem VHDL e sua utilização para síntese de circuitos na FPGA. São descritos a sua estrutura básica e os estilos de arquitetura da linguagem. Com a familiarização dos conceitos de descrição VHDL é proposto um projeto de circuito usando o estilo estrutural interconectando códigos VHDL dados.

OBJETIVO

Após a conclusão desta experiência, os seguintes tópicos devem ser conhecidos pelos alunos:

- *Familiarização com o conceito de descrição VHDL de circuitos digitais para verificar o comportamento do circuito.*
- *Familiarização do funcionamento dos circuitos na placa FPGA DE0-CV.*
- *Projeto de um circuito estrutural VHDL.*

1. PARTE EXPERIMENTAL

1.1. Linguagem de Descrição de Hardware – VHDL

O VHDL é uma linguagem de descrição de sistemas digitais. Ela nasceu durante o programa VHSIC (*Very High Speed Integrated Circuits*) do Governo dos EUA, iniciado em 1980, para desenvolver a indústria de circuitos integrados (CI). Durante o programa, logo se notou que havia a necessidade de uma linguagem padrão para descrição da estrutura e funcionalidade dos CIs. Assim, a linguagem VHSIC *Hardware Description Language* (VHDL) foi desenvolvida e logo padronizada pela IEEE (IEEE 1076-2008 – versão atual) nos EUA.

Sendo uma linguagem de descrição de hardware HDL (*Hardware Description Language*), não se deve confundir com as tradicionais linguagens de programação de software, como C ou Java. A VHDL foi projetada para cobrir todo o processo de projeto de hardware, desde o projeto da sua estrutura em níveis hierárquicos e a decomposição em subprojetos e a interconexão desses subprojetos. Pode-se especificar funções do projeto usando os comandos da linguagem, traduzi-lo em um circuito real e simulá-lo antes de ser fabricado permitindo aos projetistas comparar alternativas e testar o projeto sem os altos custos de prototipação de hardware, antes de ser programado em um dispositivo como um FPGA.

Todo projeto hierárquico de um circuito digital em VHDL possui duas partes principais: uma **entidade** e uma **arquitetura**. A **entidade** fornece um método para abstrair a funcionalidade da descrição do circuito em alto nível. Ela descreve uma lista de entradas e saídas que fazem interface do circuito com o mundo externo. A **arquitetura** descreve a implementação interna do circuito associada à **entidade**, ou seja, o que o circuito realmente faz.

Para mais detalhes veja o livro do Mealy&Tappero (2016).

Existem três abordagens diferentes (estilos) para descrever arquiteturas VHDL:

- Fluxo de dados (Data-flow);
- Comportamental (Behavioral); e
- Estrutural (Structural)

O estilo fluxo de dados descreve o circuito como uma representação concorrente de fluxo de dados pelo circuito. O circuito é descrito pelos relacionamentos entre entradas e saídas de vários componentes básicos, tais como AND, OR, XOR, etc., da linguagem VHDL. A Figura 1 apresenta um exemplo neste estilo.

Figura 1 – Estilo fluxo de dados (data-flow).

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity my_xor is
port ( A,B : in  std_logic;
       F   : out std_logic);
end my_xor;
-- architecture
architecture dataflow of my_xor is
begin
    F <= A XOR B;
end dataflow;
--
--
--
```

O estilo comportamental descreve como as saídas do circuito se comportarão diante das suas entradas. Em termos de abstração, o estilo comportamental é mais abstrato do que o estilo fluxo de dados. A Figura 2 apresenta um exemplo neste estilo.

Figura 2 – Estilo comportamental (behavioral).

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity my_xor is
port ( A,B : in  std_logic;
       F   : out std_logic);
end my_xor;
-- architecture
architecture behav of my_xor is
begin
  xor_proc: process(A,B) is
  begin
    F <= A XOR B;
  end process xor_proc;
end behav;
```

O comando "process" permite que um conjunto de instruções seja executado sequencialmente. Isto é útil quando se quer projetar circuitos sequenciais. Lembre-se de que você não está programando uma função de um software, mas sim descrevendo o comportamento de um circuito digital, mesmo que o "process" permita descrever esse comportamento por meio de instruções sequenciais. No entanto, cada comando "process" de um circuito é, por si só, um comando concorrente a outros comandos do circuito, inclusive a outros "process". Se houver mudança dos sinais presentes na lista de um comando "process", a chamada lista de sensibilidades, todos os comandos sequenciais internos são reavaliados. A Figura 3 apresenta a sintaxe do comando "process".

Figura 3 – Sintaxe do comando "process".

```
-- this is my first process
my_label: process(sensitivity_list) is
   <item_declaration>
begin
   <sequential_statements>
end process my_label;
```

O estilo estrutural é diferente das anteriores, não descreve funções lógicas usadas na operação do circuito. Serve apenas para interconectar outras arquiteturas (componentes). A Figura 4 apresenta um exemplo deste estilo que usa componentes existentes.

Figura 4 – Estilo estrutural (structural).

a) Componentes NOR e AND

```
component big_xnor
   Port ( A,B : in  std_logic;
          F : out std_logic);
end component;
```

```
component big_and3
   Port ( A,B,C : in  std_logic;
            F : out std_logic);
end component;
```

b) Estilo estrutural que usa componentes NOR e AND

```
-- library declaration
library IEEE;
use IEEE.std_logic_1164.all;
-- entity
entity my_compare is
    Port ( A_IN   : in  std_logic_vector(2 downto 0);
           B_IN   : in  std_logic_vector(2 downto 0);
           EQ_OUT : out std_logic);
end my_compare;
-- architecture
architecture ckt2 of my_compare is
   component big_xnor is
      Port ( A,B : in  std_logic;
             F : out std_logic);
   end component;

   component big_and3 is
      Port ( A,B,C : in  std_logic;
               F : out std_logic);
   end component;
   signal p1_out,p2_out,p3_out : std_logic;

begin
   x1: big_xnor port map (A_IN(2),B_IN(2),p1_out);
   x2: big_xnor port map (A_IN(1),B_IN(1),p2_out);
   x3: big_xnor port map (A_IN(0),B_IN(0),p3_out);
   a1: big_and3 port map (p1_out,p2_out,p3_out,EQ_OUT);
end ckt2;
```

1.2. Projeto de um Comparador e um Contador em VHDL

São fornecidos os projetos de um comparador binário e um contador hexadecimal em VHDL, conforme podem ser vistos na Figura 5.

Figura 5 – Comparador Binário e Contador Hexadecimal em VHDL.

a) Comparador binário

```
1  -- comparador.vhd
2  ------ comparador binario com entradas de 2 bits
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6
7  entity comparador is
8      port (
9          A, B: in std_logic_vector (1 downto 0);
10         igual: out STD_LOGIC
11     );
12 end comparador;
13
14 architecture comportamental of comparador is
15 begin
16
17     igual <= '1' when A=B else '0';
18
19 end comportamental;
```

b) Contador hexadecimal

```vhdl
-- contador.vhd
--      contador hexadecimal de 4 bits

 library IEEE;
 use IEEE.std_logic_1164.all;
 use ieee.numeric_std.all;

entity contador is
    port (clock, zera, conta, carrega: in std_logic;
          entrada: in std_logic_vector (3 downto 0);
          contagem: out std_logic_vector (3 downto 0);
          fim: out std_logic );
end contador;

architecture comportamental of contador is
  signal IQ: integer range 0 to 15;
begin

  process (clock,zera,conta,carrega,entrada,IQ)
  begin
    if zera='1' then IQ <= 0;
    elsif clock'event and clock='1' then
      if carrega='1' then
        IQ <= to_integer(unsigned(entrada));
      elsif conta='1' then
        if IQ=15 then IQ <= 0;
        else IQ <= IQ + 1;
        end if;
      else IQ <= IQ;
      end if;
    end if;
  end process;

  contagem <= std_logic_vector(to_unsigned(IQ, contagem'length));

  fim <= '1' when IQ=15 else '0';

end comportamental;
```

a) Comente os códigos fonte dos circuitos. Anexe os códigos comentados no Planejamento.

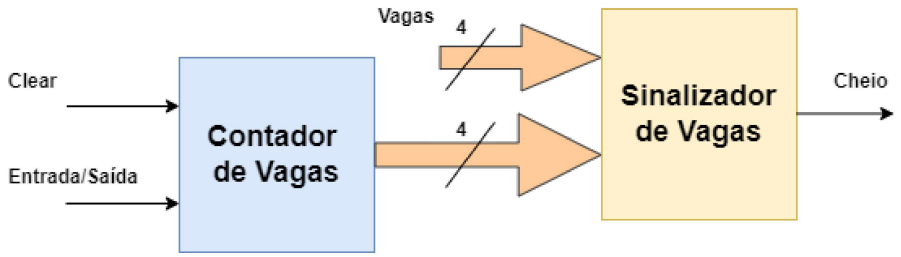
b) Simule os circuitos individualmente na ferramenta Quartus. Anexe as cartas de tempo no Planejamento.

c) Entregue o arquivo QAR (**`nome_projeto_txxbyy.qar`**) dos circuitos junto com o Planejamento, onde xx é o número da turma e yy é o número da bancada do grupo.

1.3. Projeto Estrutural em VHDL

É fornecido o projeto estrutural em VHDL que interconecta um contador e um sinalizador de vagas de estacionamento de veículos, conforme pode ser visto na Figura 6. O projeto usa um contador *up/down* binário que apresenta o número atual de veículos toda vez que um veículo entra/sai (sinal **Entrada/Saída**) do estacionamento. Quando o número de **Vagas** máximo é atingido, o sinal **Cheio** é ligado. O sinal deve ser desligado quando um veículo sai do estacionamento após este ter atingido o número máximo de vagas. O sinal **Clear**, limpa o circuito.

Figura 6 – Projeto Estrutural de um Controle de Vagas de Estacionamento..



a) Faça uma alteração no código do contador hexadecimal dado para funcionar como um contador *up/down* binário de 4 bits.
b) Integre os circuitos Contador e Sinalizador de Vagas na ferramenta Quartus. Insira o código no Planejamento.
c) Execute a função RTL Viewer da ferramenta Quartus. Verifique se o circuito está corretamente interconectado. Anexe o diagrama obtido no Planejamento.
d) Simule o circuito completo na ferramenta Quartus. Insira a Carta de Tempos do circuito no Planejamento.
e) Elabore uma Tabela de Testes do circuito com entradas, saídas e sinais intermediários. Insira a Tabela de Testes no Planejamento.
f) Entregue o arquivo QAR (`nome_projeto_txxbyy.qar`) do circuito junto com o Planejamento, onde xx é o número da turma e yy é o número da bancada do grupo.

1.4. Implementação do Circuito de Vagas de Estacionamento

a) Compile e programe o circuito Controle de Vagas de Estacionamento usando a ferramenta Quartus para a placa FPGA (DE0-CV) com Cyclone V 5CEBA4F23C7N. Faça uma Tabela de Designação de Pinais adequada (usando **chaves** e **LEDs**) para observar o funcionamento do circuito (**veja o Anexo FPGA DE0-CV – Pintable**). Insira a Tabela de Designação de Pinos no Relatório.
OBS: Não usem o botão da placa FPGA para o *Clock* do circuito. Coloquem a entrada em um pino da interface GPIO. Será usado o dispositivo Analog Discovery para simular o botão da placa.
b) Faça testes no circuito, seguindo a Tabela de Testes planejada. Caso algum teste não seja satisfatório, insira sinais de depuração para detectar o problema. Em seguida, corrija o problema e teste o circuito novamente. Comente os resultados no Relatório.

1.5. Desafio (Opcional)

O professor irá propor um desafio sobre esta experiência.

2. BIBLIOGRAFIA

- ALMEIDA, F.V. de; SATO, L.M.; MIDORIKAWA, E.T. **Tutorial para criação de circuitos digitais em VHDL no Quartus Prime 16.1**. Apostila de Laboratório Digital. Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP. Edição de 2017.
- MIDORIKAWA, E. T. **Primeiro Contato com Circuitos Digitais em VHDL**. Apostila de Laboratório Digital. Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP. Edição de 2019.
- ALTERA. **DE0-CV User Manual**. 2015.
- ALTERA. **Quartus Prime Introduction Using VHDL Designs.** 2016.
- ALTERA. **Quartus Prime Introduction to Simulation of VHDL Designs.** 2016.
- ALTERA. **Quartus Prime Introduction Using Schematic Designs.** 2016.
- D'AMORE, R. **VHDL - descrição e síntese de circuitos digitais**. 2ª edição, LTC, 2012
- PCS-EPUSP. **Calculadora simples em VHDL**. Apostila de Laboratório Digital. 2015.
- TOCCI, R.J. & WIDMER,N.S. **Sistemas digitais**: princípios e aplicações. 11ª ed., Prentice-Hall, 2011.

- WAKERLY, John F. **Digital Design Principles & Practices**. 4th edition, Prentice Hall, 2006.
- ASHENDEN, P. J. The VHDL Cookbook. 1st edition, University of Adelaide, Australia, July. 1990.
- MEALY, B.; TAPPERO F. **Free Range VHDL**. freerangefactory.org. 2016.
- NEEMANN, H. DIGITAL: https://github.com/hneemann/Digital consultado em Abril, 2021.

3. RECURSOS NECESSÁRIOS

- 1 Computador pessoal.
- 1 Placa FPGA DE0-CV da Altera com o dispositivo Cyclone V 5CEBA4F23C7N.
- 1 Dispositivo Analog Discovery.
- 1 Ferramenta Intel Quartus Prime 16.1 da Altera.
- 1 Ferramenta Waveforms do Analog Discovery.

ANEXOS

- Tutorial para Criação de Circuitos Digitais em VHDL
- FPGA DE0-CV - Pintable

Histórico de Revisões

Profs. Kechi Hirama, Jorge Rady de Almeida Júnior, Sérgio Roberto de Mello Canovas – versão 2020

Profs. Kechi Hirama, Jorge Kinoshita – versão 2021

Profs. Kechi Hirama, Jorge Rady de Almeida Júnior - versão 2022