*Ministério da Educação*
**Universidade Tecnológica Federal do Paraná**
Câmpus Apucarana
Bacharelado em Engenharia de Computação

Universidade Tecnológica Federal do Paraná

Engenharia de Computação

# PROJETO DETECTOR DE MÍNIMO E MÁXIMO

Maria Eduarda Pedroso

Professor orientador: Marcelo de Oliveira

Apucarana

Outubro/2023

# SUMÁRIO

## 1. RESUMO

Este trabalho propõe o desenvolvimento de um circuito digital capaz de detectar valores mínimos e máximos em um conjunto de dados representados como números unsigned. A estratégia emprega a linguagem de descrição de hardware VHDL, fazendo uso de estruturas como package, procedure, generic e type. Através do pacote definido, foi possível estabelecer um vetor de unsigned com flexibilidade no número de bits e quantidade de valores. Uma procedure foi estabelecida para rastrear o vetor de dados, identificando e atualizando os valores mínimos e máximos conforme necessário. A solução oferece adaptabilidade a diferentes cenários e aplicações, representando uma abordagem eficaz para sistemas que demandam operações de comparação em arrays de dados.

Palavras-chave: VHDL, detector de valores, unsigned, procedure, circuito digital.

## 2. INTRODUÇÃO

Em um mundo cada vez mais dominado por sistemas digitais e eletrônicos, a habilidade de processar, interpretar e agir sobre conjuntos de dados em tempo real é de suma importância. Circuito digitais, sendo a espinha dorsal de tais sistemas, precisam ser precisos, eficientes e adaptáveis para lidar com diferentes tamanhos e tipos de dados. Dentro deste contexto, o presente trabalho se concentra no desenvolvimento de um circuito digital capaz de identificar valores mínimos e máximos de um conjunto de dados representados como números unsigned.

A linguagem VHDL, acrônimo para VHSIC Hardware Description Language (ou Linguagem de Descrição de Hardware para Circuitos Integrados de Altíssima Velocidade), foi escolhida como a ferramenta principal de desenvolvimento devido à sua versatilidade, precisão e amplo uso na indústria e academia. Esta linguagem permite não apenas a descrição detalhada de hardware, mas também a simulação e teste de tais descrições, garantindo assim um processo de design mais confiável.

O núcleo teórico deste trabalho reside na arquitetura de circuitos comparadores e sua implementação eficiente. Comparadores são fundamentais em sistemas digitais e encontram aplicação em uma variedade de cenários, desde simples decisões lógicas até operações mais complexas como as realizadas em unidades aritméticas e lógicas (ALUs) de microprocessadores. Ao focar no desafio de identificar valores mínimos e máximos em um vetor de números unsigned, este projeto busca explorar e destacar os princípios subjacentes de comparação binária e operações aritméticas em sistemas digitais.

Além disso, será dada ênfase à modularização e reutilização de código em VHDL, utilizando estruturas como package e procedure, bem como a adaptabilidade proporcionada por estruturas como generic e type. Estas abordagens não apenas facilitam o design e a manutenção do código, mas também proporcionam uma plataforma para a extensibilidade e adaptação a diferentes aplicações e requisitos.

Em resumo, este projeto visa não apenas atender a um objetivo funcional específico, mas também servir como uma exploração profunda da teoria e prática por trás da implementação de circuitos comparadores eficientes em VHDL.

## 2.1. Objetivos

O presente trabalho tem como objetivo:

- Desenvolvimento de Circuitos: Desenvolver um circuito digital que consiga processar uma sequência de números unsigned e identificar os valores mínimo e máximo dentro desta sequência.
- Aplicação de VHDL: Utilizar a linguagem VHDL, uma linguagem de descrição de hardware, como ferramenta principal para descrever e implementar o circuito proposto.
- Flexibilidade e Adaptação: Projetar o circuito de maneira que ele possa ser facilmente adaptado para diferentes tamanhos de vetor e diferentes bit-larguras de números unsigned. Para isso, a utilização de estruturas como generic e type será crucial.
- Otimização e Eficiência: Embora não tenha sido explicitamente mencionado, um objetivo implícito poderia ser garantir que o circuito funcione de forma eficiente, consumindo o mínimo de recursos e oferecendo respostas em tempo hábil.

- Aplicação de Estruturas de VHDL: Explorar e aplicar estruturas avançadas de VHDL, como package, procedure, e outras, para modularizar o código e torná-lo reutilizável.

## 3. METODOLOGIA

O presente trabalho foi desenvolvido utilizando os conceitos e técnicas estudados durante as aulas da disciplina. Bem como foi empregado o uso do software Quartus, de modo a facilitar e simular nossa lógica.

### 3.1. Funcionalidades do projeto

#### 3.1.1. Package

Um package em VHDL é uma coleção de definições e declarações que podem ser reutilizadas em vários pontos do código. No nosso caso, temos um pacote chamado unsigned_pkg, que contém uma definição de tipo (unsigned_vector) e uma procedure (encontrar_min_max). Abaixo podemos ver esse trecho do código e a chamada dele em outro arquivo

```
use work.unsigned_pkg.ALL;
```

```
package unsigned_pkg is
   type unsigned_vector is array (natural range <>) of unsigned;
   procedure encontrar_min_max(values: in unsigned_vector;
                               NUM_VALUES, BIT_SIZE: in positive;
                               signal min_value, max_value: out unsigned);
end package unsigned_pkg;
```

No codigo acima, unsigned_pkg é o nome do pacote, que contém a definição do tipo unsigned_vector, que é um vetor de sinais não sinalizados

(unsigned). Além disso, há uma procedure chamada encontrar_min_max, que será explicada em mais detalhes abaixo.

### 3.1.2. Procedure

Uma procedure em VHDL é um bloco de código reutilizável que realiza uma tarefa específica. No código que foi feito, a procedure encontrar_min_max encontra os valores mínimo e máximo em um vetor de sinais não sinalizados.

```
procedure encontrar_min_max(values: in unsigned_vector;
                            NUM_VALUES, BIT_SIZE: in positive;
                            signal min_value, max_value: out unsigned) is
    variable temp_min, temp_max: unsigned(BIT_SIZE - 1 downto 0);
begin
    temp_min := values(0);
    temp_max := values(0);

    for i in 0 to NUM_VALUES - 1 loop
        if values(i) < temp_min then
            temp_min := values(i);
        end if;

        if values(i) > temp_max then
            temp_max := values(i);
        end if;
    end loop;

    min_value <= temp_min;
    max_value <= temp_max;
end procedure;
```

Esta procedure acima, values é o vetor de sinais não sinalizados de entrada, e min_value e max_value são sinais de saída que armazenam os valores mínimo e máximo, respectivamente. A procedure utiliza um loop for para percorrer o vetor e encontrar os valores que são o máximo e o mínimo.

### 3.1.3. Generic

A palavra-chave generic é usada para definir parâmetros que podem ser ajustados ou personalizados quando uma entidade é instanciada. No código, temos dois exemplos de utilização de generic.

```
generic (
   BIT_SIZE: positive := 4;  -- Default bit size
   NUM_VALUES: positive := 4 -- Default number of values
);
```

Esses generics (BIT_SIZE e NUM_VALUES) permite que ajustemos o tamanho dos bits e o número de valores de entrada quando instanciar a entidade aula230830. Isso proporciona flexibilidade para adaptar a funcionalidade conforme necessário sem modificar o código interno.

### 3.1.4. Type

A palavra-chave type é usada para definir um novo tipo de dado. No código, utilizamos a declaração type unsigned_vector para criar um novo tipo de vetor que armazena sinais não sinalizados.

```
type unsigned_vector is array (natural range <>) of unsigned;
```

Acima, unsigned_vector é um tipo que representa um vetor de sinais não sinalizados (unsigned). Essa definição facilita a leitura e compreensão do código, tornando-o mais modular e reutilizável.

## 3.2. Código VHDL

Abaixo temos o código inteiro que foi implementado para a atividade.

```
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.NUMERIC_STD.ALL;
--TODO: ARRUMAR CLOCK
----------------------------------------------------------------------------------------------------
entity aula230830 is
                    generic(
                    --f_clk : integer := f_clk'high; -- variavel contadora
                    f_clk : integer := 50_000_000
                    );
  ----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.unsigned_pkg.ALL;
----------------------------------------------------------------------------------------------------
entity aula230830 is

  generic (
    BIT_SIZE: positive := 4;  -- Default bit size
    NUM_VALUES: positive := 4 -- Default number of values
  );
  port (
--      values: in unsigned_vector (0 to NUM_VALUES-1)(BIT_SIZE-1 downto 0);
                    in1, in2, in3, in4: in unsigned(BIT_SIZE-1 downto 0);
    min_value: out unsigned(BIT_SIZE-1 downto 0);
    max_value: out unsigned(BIT_SIZE-1 downto 0)
  );
end entity aula230830;
----------------------------------------------------------------------------------------------------
architecture Behavioral of aula230830 is
          signal values: unsigned_vector (0 to NUM_VALUES-1)(BIT_SIZE-1 downto 0);
begin
  values <= (in1,in2,in3,in4);
 encontrar_min_max(values,NUM_VALUES,BIT_SIZE,min_value,max_value);
end architecture Behavioral;
```

```
library IEEE;
```

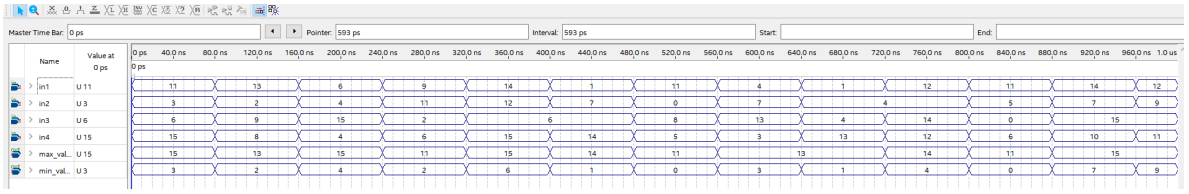
```
use ieee.numeric_std.all;

package unsigned_pkg is
  type unsigned_vector is array (natural range <>) of unsigned;
  procedure encontrar_min_max(values: in unsigned_vector;
                              NUM_VALUES, BIT_SIZE: in positive;
                              signal min_value, max_value: out unsigned);
end package unsigned_pkg;

package body unsigned_pkg is
  procedure encontrar_min_max(values: in unsigned_vector;
                              NUM_VALUES, BIT_SIZE: in positive;
                              signal min_value, max_value: out unsigned) is
    variable temp_min, temp_max: unsigned(BIT_SIZE - 1 downto 0);
  begin
                               temp_min := values(0);
                               temp_max := values(0);

    for i in 0 to NUM_VALUES - 1 loop
      if values(i) < temp_min then
        temp_min := values(i);
      end if;

      if values(i) > temp_max then
        temp_max := values(i);
      end if;
    end loop;

    min_value <= temp_min;
    max_value <= temp_max;
  end procedure;

end package body unsigned_pkg;
```

## 4. RESULTADOS E DISCUSSÃO

Com a implementação funcionando utilizamos da simulação no quartus para testar nosso código, abaixo temos duas imagens mostrando as saídas simuladas.

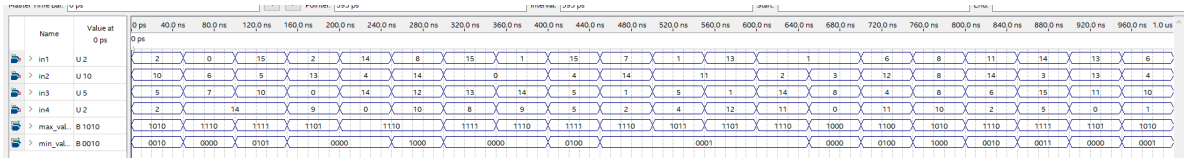
Podemos concluir com a figura 1 e 2 que nosso código funciona de maneira eficaz, sendo o valor da saída mínima o menor valor das entradas e da saída máxima o maior valor dentre as entradas.

Figura 1 - Simulação 1.



Fonte: Autoria Própria

Figura 2 - Simulação 2.



Fonte: Autoria Própria

## 5. CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, investigamos a criação e a operação de um detector de valores mínimo e máximo utilizando a linguagem de descrição de hardware VHDL. O desafio central era identificar, dentro de um conjunto de números unsigned, os valores que representavam os extremos do conjunto, sejam eles o menor ou o maior valor.

Através da implementação proposta, foi possível observar a robustez e adaptabilidade proporcionadas pela linguagem VHDL. A utilização de estruturas como package, procedure, generic e type não apenas facilitou o design do circuito, mas também ofereceu um meio de tornar o código modular, reutilizável e facilmente adaptável para diferentes aplicações e requisitos.

É digno de nota que, em sistemas digitais, a eficiência do design não é medida apenas pela capacidade de atender a um requisito funcional. A otimização, a modularidade e a capacidade de adaptar e reutilizar o código em diferentes cenários

são aspectos cruciais que determinam a qualidade e a viabilidade de um projeto. A solução apresentada neste trabalho evidencia tais características e serve como um ponto de referência para futuras implementações ou estudos relacionados.

Por fim, este estudo não apenas cumpriu seu propósito inicial, mas também reforçou a importância da abordagem teórica e prática equilibrada no design de circuitos digitais. A teoria por trás das operações de comparação e as técnicas aplicadas para implementar tal funcionalidade em VHDL são, sem dúvida, fundamentais para qualquer engenheiro ou entusiasta da área.

O campo da eletrônica digital continua a ser um domínio empolgante e desafiador, e este trabalho serve como um lembrete do potencial e das possibilidades que ainda esperam ser exploradas.