

## **Projeto Final - Calculadora**

Aluno: Deivid da Silva Galvão RA: 2408740  
Aluno: João Vitor Nakahodo Yoshida RA: 2419904  
Professor orientador: Marcelo de Oliveira

## **Relatório**

Relatório do Trabalho Prático Disciplinar apresentado como requisito parcial à obtenção de nota na disciplina de Lógica Reconfigurável do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Aluno: Deivid da Silva Galvão RA: 2408740

Aluno: João Vitor Nakahodo Yoshida RA: 2419904

Professor orientador: Marcelo de Oliveira

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Implementação no Quartus Prime</b>	<b>1</b>
2.1	Código VHDL: . . . . .	1
2.2	Explicação do Código: . . . . .	3
2.3	Pin Planner . . . . .	4
<b>3</b>	<b>Resultados Obtidos</b>	<b>6</b>
3.1	Na Placa . . . . .	6
<b>4</b>	<b>Conclusão</b>	<b>8</b>
4.1	Sugestões de melhorias . . . . .	8

# 1 Introdução

O projeto final visa a implementação de uma calculadora utilizando VHDL e um display de 7 segmentos. O objetivo é permitir a seleção de números e operações através de switches, e confirmar a operação por meio de um botão. O resultado da operação será exibido no display de 7 segmentos.

## 2 Implementação no Quartus Prime

### 2.1 Código VHDL:

---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.Bin7seg.ALL;

entity progF is
    Port (
        sw_a : in std_logic_vector (3 downto 0);
        sw_b : in std_logic_vector (3 downto 0);
        op_sel : in std_logic_vector (1 downto 0);
        confirm : in std_logic;
        seg_a : out std_logic_vector (7 downto 0);
        seg_b : out std_logic_vector (7 downto 0);
        seg_singnal : out std_logic_vector (7 downto 0);
        seg_result1 : out std_logic_vector (7 downto 0);
        seg_result : out std_logic_vector (7 downto 0);
        seg_result2 : out std_logic_vector (7 downto 0)
    );
end progF;

architecture progF of progF is
    signal a, b : unsigned (3 downto 0);
    signal result : unsigned (7 downto 0);
    signal tens, ones : unsigned (3 downto 0);

begin
    process(confirm)
        variable temp_result : unsigned(7 downto 0);
        variable temp_tens, temp_ones : unsigned(3 downto 0);
    begin
        if rising_edge(confirm) then
            a <= unsigned(sw_a);
            b <= unsigned(sw_b);

            case op_sel is
```

```

when "00" => — add
    temp_result := resize(a + b, 8);
    seg_signal <= "11000000"; — "+" no display
    seg_result2 <= "11111111";
when "01" => — Sub
    if a >= b then
        temp_result := resize(a - b, 8);
        seg_result2 <= "11111111";
    else
        temp_result := resize(b - a, 8);
        seg_result2 <= "10111111";
    end if;
    seg_signal <= "11111001"; — "-" no display
when "10" => — Mult
    temp_result := resize(a * b, 8);
    seg_signal <= "00100100"; — "x" no display
    seg_result2 <= "11111111";
when "11" => — Div
    if b /= 0 then
        temp_result := resize(a / b, 8);
    else
        temp_result := (others => '0');
    end if;
    seg_signal <= "00110000"; — "/" no display
    seg_result2 <= "11111111";
when others =>
    temp_result := (others => '0');
end case;

— Conversao para BCD
temp_ones := resize(temp_result mod 10, 4);
temp_tens := resize((temp_result / 10), 4);

— Atribuir valores aos sinais
ones <= temp_ones;
tens <= temp_tens;
end if;
end process;

— Atualiza os displays de 7 segmentos
seg_a <= bin_to_7seg(std_logic_vector(a));
seg_b <= bin_to_7seg(std_logic_vector(b));
seg_result1 <= bin_to_7seg(std_logic_vector(ones)); — unidade
seg_result <= bin_to_7seg(std_logic_vector(tens)); — dezena
end progF;

```

---



---

```
library IEEE;
```

```

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

package Bin7seg is
    function bin_to_7seg(bin : std_logic_vector (3 downto 0)) ...
    ... return std_logic_vector;
end package Bin7seg ;

package body Bin7seg is
    function bin_to_7seg(bin : std_logic_vector (3 downto 0)) ...
    ... return std_logic_vector is
    begin
        case bin is
            when "0000" => return "11000000"; — 0
            when "0001" => return "11111001"; — 1
            when "0010" => return "10100100"; — 2
            when "0011" => return "10110000"; — 3
            when "0100" => return "10011001"; — 4
            when "0101" => return "10010010"; — 5
            when "0110" => return "10000010"; — 6
            when "0111" => return "11111000"; — 7
            when "1000" => return "10000000"; — 8
            when "1001" => return "10010000"; — 9
            when others => return "11111111"; — Desligado
        end case;
    end function;
end package body Bin7seg ;

```

---

## 2.2 Explicação do Código:

O código VHDL é implementado dentro de uma entidade chamada ‘progF’ e utiliza bibliotecas da IEEE para operações lógicas e aritméticas. A entidade ‘progF’ possui sinais de entrada para 2 números, um seletor e um ”confirmador”, e sinais de saída para os 6 displays de 7 segmentos, representando o primeiro numero, o segundo numero, o operador, o sinal e os dois displays de resultado (dezena e unidade).

Dentro da arquitetura ‘progF’, é declarado um tipo unsigned (números inteiros sem sinal (vetor de bits)) de 4 bits para representar o sinal do primeiro e segundo número e das dezenas e unidades do resultado, e também um tipo unsigned para de 8 bits para o resultado.

O processo principal é sensível ao clique do botão. Ele ultiliza a borda de subida para salvar os valores dos *switches*, sendo os 4 mais a direita o num1 e os 4 posteriores para esquerda são o num2, e a operação escolhida por meio dos 2 *switches* mais a esquerda.

A operação de adição ocorre quando o seletor está em ’00’, a subtração é realizada quando o seletor está em ’01’, a multiplicação é executada quando o seletor está em ’10’ e, por fim, a divisão acontece quando o seletor está em ’11’.

Para fazer a conversão binário para decimal do resultado foi usada a seguinte lógica: A dezena é o resultado dividido por 10 enquanto que as unidades são o resto dessa divisão e a unidade e dezena são redimensionadas para manter 4 bits de tamanho.

Por fim é chamada a função que converte binário para o display de 7 segmentos por meio do package "use work.Bin7seg.ALL;" e após isso é mostrado tudo nos displays.

## 2.3 Pin Planner

Após a finalização do código VHDL foi feito o *pin planner* do projeto, que acabou ficando bem grande em função do uso de vários displays de 7 segmentos.

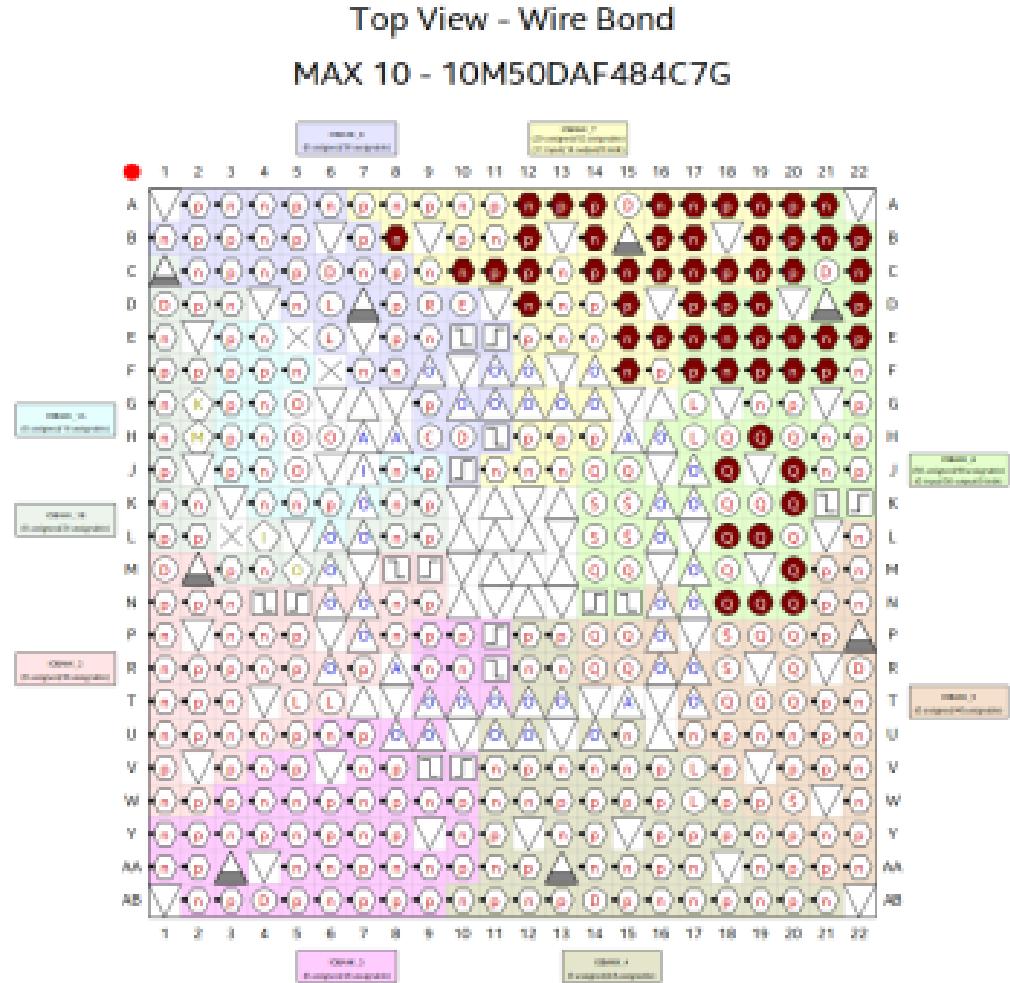


Figura 1: Pin Planner

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
in  confirm	Input	PIN_B8	7	B7_N0	PIN_B8	2.5 V		12mA (default)			
in  op_sel[1]	Input	PIN_F15	7	B7_N0	PIN_F15	2.5 V		12mA (default)			
in  op_sel[0]	Input	PIN_B14	7	B7_N0	PIN_B14	2.5 V		12mA (default)			
out  seg_a[7]	Output	PIN_L19	6	B6_N0	PIN_L19	2.5 V		12mA (default)	2 (default)		
out  seg_a[6]	Output	PIN_N20	6	B6_N0	PIN_N20	2.5 V		12mA (default)	2 (default)		
out  seg_a[5]	Output	PIN_N19	6	B6_N0	PIN_N19	2.5 V		12mA (default)	2 (default)		
out  seg_a[4]	Output	PIN_M20	6	B6_N0	PIN_M20	2.5 V		12mA (default)	2 (default)		
out  seg_a[3]	Output	PIN_N18	6	B6_N0	PIN_N18	2.5 V		12mA (default)	2 (default)		
out  seg_a[2]	Output	PIN_L18	6	B6_N0	PIN_L18	2.5 V		12mA (default)	2 (default)		
out  seg_a[1]	Output	PIN_K20	6	B6_N0	PIN_K20	2.5 V		12mA (default)	2 (default)		
out  seg_a[0]	Output	PIN_J20	6	B6_N0	PIN_J20	2.5 V		12mA (default)	2 (default)		
out  seg_b[7]	Output	PIN_D22	6	B6_N0	PIN_D22	2.5 V		12mA (default)	2 (default)		
out  seg_b[6]	Output	PIN_E17	6	B6_N0	PIN_E17	2.5 V		12mA (default)	2 (default)		
out  seg_b[5]	Output	PIN_D19	6	B6_N0	PIN_D19	2.5 V		12mA (default)	2 (default)		
out  seg_b[4]	Output	PIN_C20	6	B6_N0	PIN_C20	2.5 V		12mA (default)	2 (default)		
out  seg_b[3]	Output	PIN_C19	7	B7_N0	PIN_C19	2.5 V		12mA (default)	2 (default)		
out  seg_b[2]	Output	PIN_E21	6	B6_N0	PIN_E21	2.5 V		12mA (default)	2 (default)		
out  seg_b[1]	Output	PIN_E22	6	B6_N0	PIN_E22	2.5 V		12mA (default)	2 (default)		
out  seg_b[0]	Output	PIN_F21	6	B6_N0	PIN_F21	2.5 V		12mA (default)	2 (default)		
out  seg_result[7]	Output	PIN_A19	7	B7_N0	PIN_A19	2.5 V		12mA (default)	2 (default)		
out  seg_result[6]	Output	PIN_B22	6	B6_N0	PIN_B22	2.5 V		12mA (default)	2 (default)		
out  seg_result[5]	Output	PIN_C22	6	B6_N0	PIN_C22	2.5 V		12mA (default)	2 (default)		
out  seg_result[4]	Output	PIN_B21	6	B6_N0	PIN_B21	2.5 V		12mA (default)	2 (default)		
out  seg_result[3]	Output	PIN_A21	6	B6_N0	PIN_A21	2.5 V		12mA (default)	2 (default)		
out  seg_result[2]	Output	PIN_B19	7	B7_N0	PIN_B19	2.5 V		12mA (default)	2 (default)		
out  seg_result[1]	Output	PIN_A20	7	B7_N0	PIN_A20	2.5 V		12mA (default)	2 (default)		
out  seg_result[0]	Output	PIN_B20	6	B6_N0	PIN_B20	2.5 V		12mA (default)	2 (default)		
out  seg_result[7]	Output	PIN_A16	7	B7_N0	PIN_A16	2.5 V		12mA (default)	2 (default)		
out  seg_result[6]	Output	PIN_B17	7	B7_N0	PIN_B17	2.5 V		12mA (default)	2 (default)		
out  seg_result[5]	Output	PIN_A18	7	B7_N0	PIN_A18	2.5 V		12mA (default)	2 (default)		
out  seg_result[4]	Output	PIN_A17	7	B7_N0	PIN_A17	2.5 V		12mA (default)	2 (default)		
out  seg_result[3]	Output	PIN_B16	7	B7_N0	PIN_B16	2.5 V		12mA (default)	2 (default)		
out  seg_result[2]	Output	PIN_E18	6	B6_N0	PIN_E18	2.5 V		12mA (default)	2 (default)		
out  seg_result[1]	Output	PIN_D18	6	B6_N0	PIN_D18	2.5 V		12mA (default)	2 (default)		
out  seg_result[0]	Output	PIN_C18	7	B7_N0	PIN_C18	2.5 V		12mA (default)	2 (default)		
out  seg_result[7]	Output	PIN_D15	7	B7_N0	PIN_D15	2.5 V		12mA (default)	2 (default)		
out  seg_result[6]	Output	PIN_C17	7	B7_N0	PIN_C17	2.5 V		12mA (default)	2 (default)		
out  seg_result[5]	Output	PIN_D17	7	B7_N0	PIN_D17	2.5 V		12mA (default)	2 (default)		
out  seg_result[4]	Output	PIN_E16	7	B7_N0	PIN_E16	2.5 V		12mA (default)	2 (default)		
out  seg_result[3]	Output	PIN_C16	7	B7_N0	PIN_C16	2.5 V		12mA (default)	2 (default)		
out  seg_result[2]	Output	PIN_C15	7	B7_N0	PIN_C15	2.5 V		12mA (default)	2 (default)		
out  seg_result[1]	Output	PIN_E15	7	B7_N0	PIN_E15	2.5 V		12mA (default)	2 (default)		
out  seg_result[0]	Output	PIN_C14	7	B7_N0	PIN_C14	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[7]	Output	PIN_F17	6	B6_N0	PIN_F17	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[6]	Output	PIN_F20	6	B6_N0	PIN_F20	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[5]	Output	PIN_F19	6	B6_N0	PIN_F19	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[4]	Output	PIN_H19	6	B6_N0	PIN_H19	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[3]	Output	PIN_J18	6	B6_N0	PIN_J18	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[2]	Output	PIN_E19	6	B6_N0	PIN_E19	2.5 V		12mA (default)	2 (default)		
out  seg_singnal[1]	Output	PIN_E20	6	B6_N0	PIN_E20	2.5 V		12mA (default)	2 (default)		

Figura 2: Pin Planner

out  seg_singnal[0]	Output	PIN_F18	6	B6_N0	PIN_F18	2.5 V		12mA (default)	2 (default)		
in  sw_a[3]	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V		12mA (default)			
in  sw_a[2]	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V		12mA (default)			
in  sw_a[1]	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V		12mA (default)			
in  sw_a[0]	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)			
in  sw_b[3]	Input	PIN_A14	7	B7_N0	PIN_A14	2.5 V		12mA (default)			
in  sw_b[2]	Input	PIN_A13	7	B7_N0	PIN_A13	2.5 V		12mA (default)			
in  sw_b[1]	Input	PIN_B12	7	B7_N0	PIN_B12	2.5 V		12mA (default)			
in  sw_b[0]	Input	PIN_A12	7	B7_N0	PIN_A12	2.5 V		12mA (default)			

Figura 3: Pin Planner

### 3 Resultados Obtidos

#### 3.1 Na Placa

Com o *pin planner* finalizado o código foi compilado novamente e enviado para a placa para a realização dos teste e verificação do seu funcionamento. Pode-se observar que foram obtidos resultados satisfatórios. No exemplo abaixo, foi feita a soma de 8 com 3 sendo o resultado mostrado no display 2 e 1 como 11.

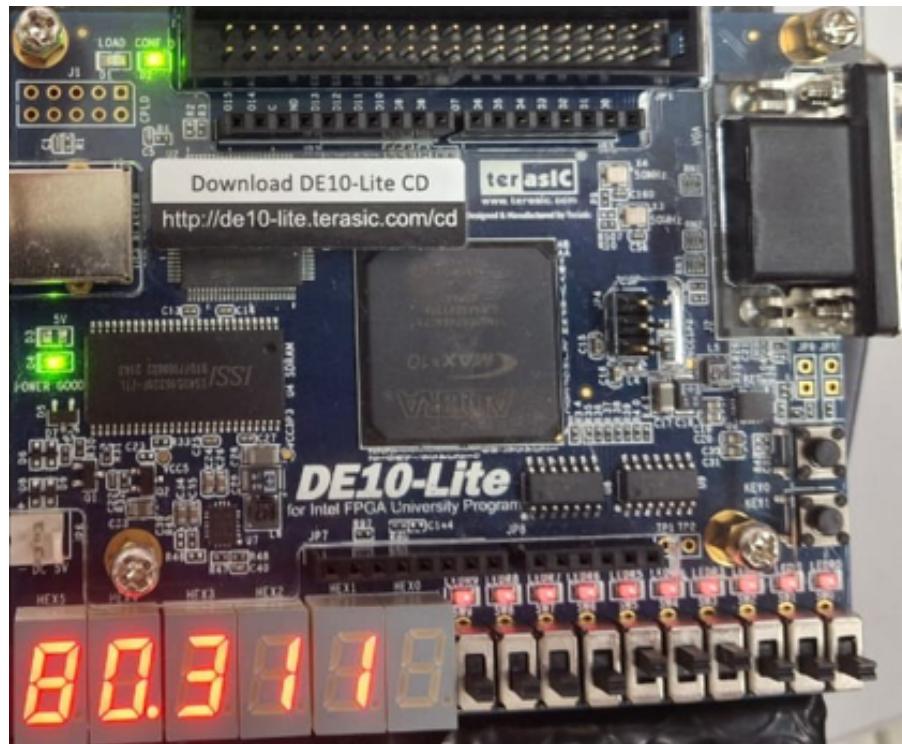


Figura 4: Teste de soma na placa

Após isso foi realizado um teste de subtração com o intuito de verificar o funcionamento de números negativos, além da subtração em si, o resultado é mostrado na figura 6.

Logo em seguida foi realizado um teste de divisão, também com resultado correto, onde foi dividido "8/2" e o resultado obtido foi "04", resultado é mostrado na figura 7.

E por fim foi realizado o teste de multiplicação onde foi multiplicado 8 com 7 e o resultado foi 56 como mostra a figura 8



Figura 5: Teste de subtração na placa



Figura 6: Teste de divisão na placa



Figura 7: Teste de multiplicação na placa

## 4 Conclusão

O projeto desenvolvido implementa uma calculadora em VHDL, utilizando *package* para modularização e displays de 7 segmentos para exibição dos resultados. A estrutura proposta permite realizar operações básicas, como adição, subtração, multiplicação e divisão, garantindo uma interface clara e intuitiva para o usuário.

### 4.1 Sugestões de melhorias

Ao invés de representar o sinal no display, pode ser interessante utilizar um LED para essa finalidade. Podendo assim, usar o último display de 7 segmentos para representar a centena, caso o resultado seja maior que 99. Além disso, melhorias podem incluir a multiplexação dos *switches* para reduzir o número de pinos utilizados e a implementação de alguma forma sinalização de erro para casos com divisão pro 0, por exemplo.