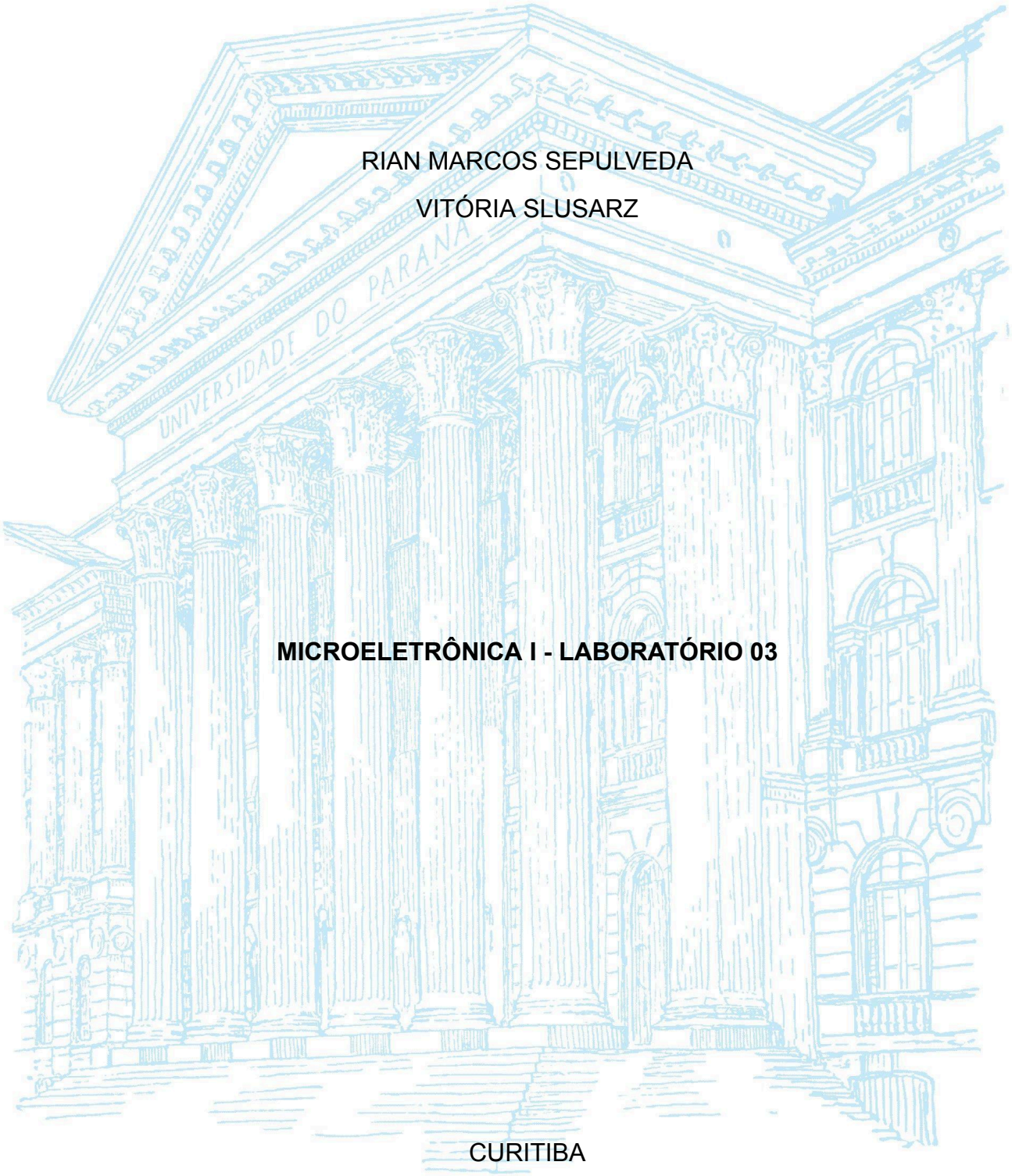
UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
SETOR DE TECNOLOGIA



RIAN MARCOS SEPULVEDA
VITÓRIA SLUSARZ

MICROELETRÔNICA I - LABORATÓRIO 03

CURITIBA
2024

RIAN MARCOS SEPULVEDA
VITÓRIA SLUSARZ

MICROELETRÔNICA I - LABORATÓRIO 03

Trabalho apresentado no curso de Engenharia Elétrica para a disciplina de Microeletrônica I solicitado como requisito de avaliação parcial da disciplina.
Orientador: Prof. Dr. Sibilla Batista da Luz França.

CURITIBA

2024

RESUMO

O presente relatório aborda os dois experimentos realizados no terceiro laboratório da disciplina de Microeletrônica I, sendo eles um temporizador e um contador que contam de 0 a 9. Antes dos testes realizados em sala de aula, todos os esquemáticos foram previamente simulados na plataforma Xilinx ISE e implementados na placa XC3S500E

1 INTRODUÇÃO

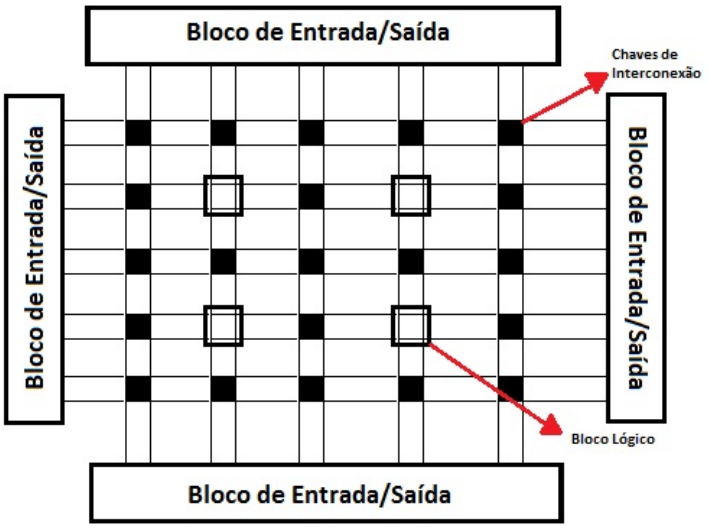
Neste relatório, são apresentados os resultados de um laboratório prático dedicado à implementação de um temporizador e de um contador de 0 a 9 utilizando linguagem de descrição de hardware VHDL e a placa FPGA XC3S500E. Além dos resultados, são detalhadas todas as etapas de cada projeto, incluindo simulação e teste dos circuitos.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo visa oferecer uma base conceitual para compreender os princípios relacionados aos temas abordados no laboratório em questão. Dessa forma, serão explorados fundamentos teóricos dos FPGA (Field-Programmable Gate Arrays) e da linguagem de descrição de hardware VHDL (VHSIC Hardware Description Language), bem como também conceitos adicionais utilizados em cada um dos projetos realizados.

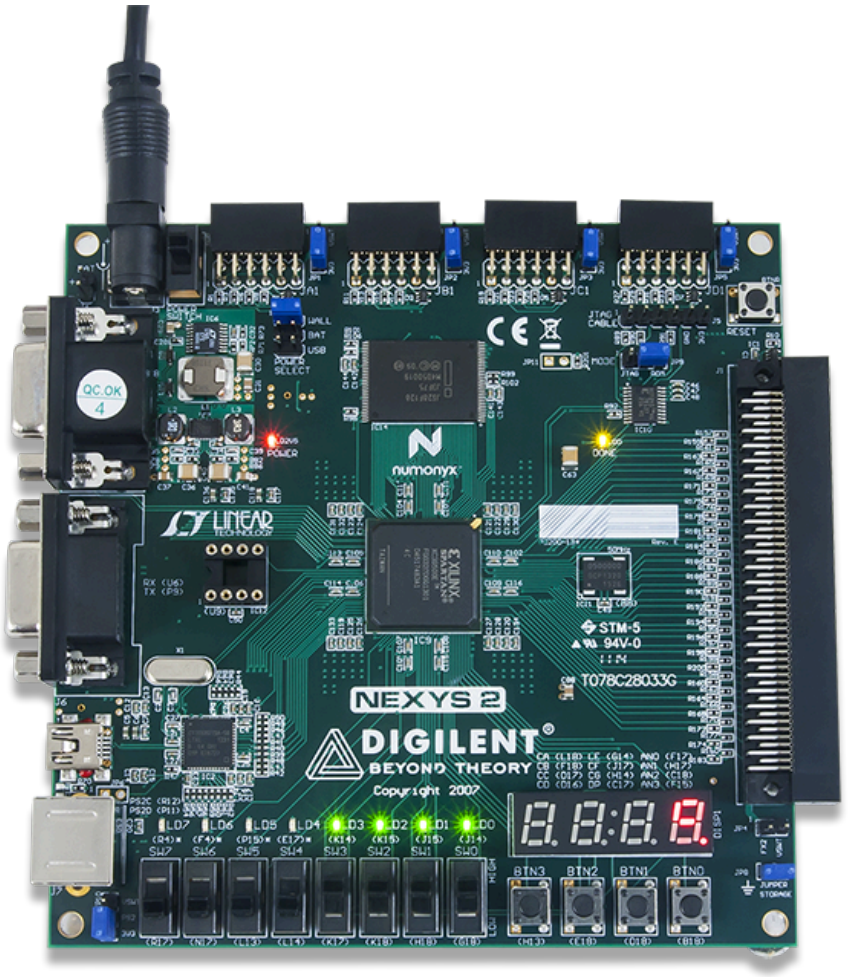
2.1 FPGA - XC3S500E:

O XC3S500E é um FPGA (Field-Programmable Gate Arrays) que pertence à família Spartan-3E da Xilinx e é conhecido por seu custo-benefício. Os arranjos de porta programável em campo (FPGAs) são dispositivos semicondutores que consistem em uma matriz de blocos lógicos configuráveis, blocos de entrada/saída, e interconexões programáveis.



O equipamento utilizado na disciplina conta com quinhentas mil portas lógicas equivalentes (LEs), conferindo a ele uma capacidade considerável para a implementação de circuitos digitais mais complexos. Essa capacidade pode ser traduzida em até 192 blocos lógicos configuráveis (CLBs) e até 4 blocos DSP (Digital Signal Processing), fazendo com que o XC3S500E seja adequado para aplicações

que requerem um processamento intensivo de sinais digitais. Além disso, o XC3S500E possui uma variedade de recursos adicionais que fazem com que ele seja versátil em diferentes contextos de aplicação, como por exemplo sua memória embutida, que proporciona armazenamento local para dados e configurações.



Sendo assim, para que o dispositivo fosse programado e configurado, foi utilizado o software Xilinx ISE (Integrated Software Environment), que permite a descrição do circuito em uma linguagem de descrição de hardware, além da síntese e implementação do design na placa.

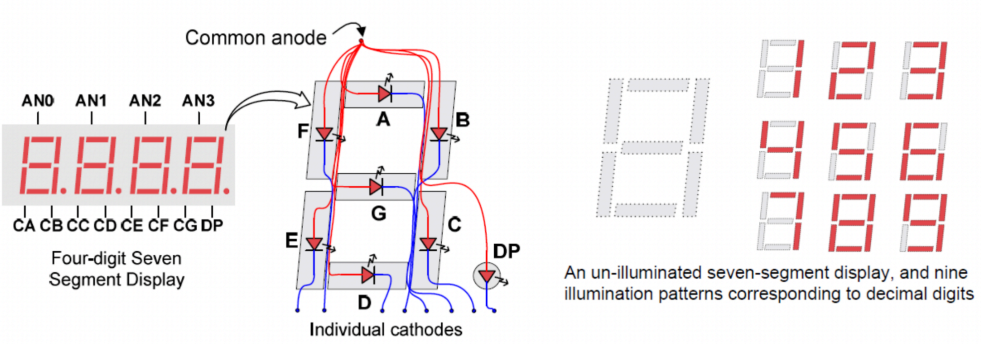
2.2 VHDL:

A linguagem VHDL (VHSIC Hardware Description Language) é uma linguagem de descrição de hardware, sendo amplamente utilizada para realizar o design de circuitos integrados e FPGAs. Com ela é possível modelar desde

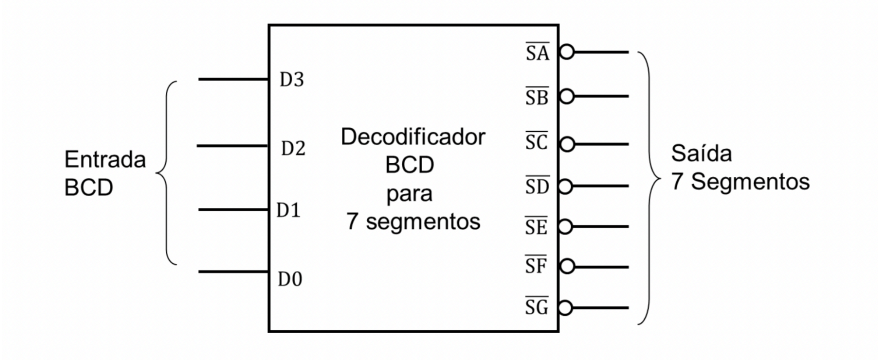
pequenos componentes lógicos até sistemas mais completos, especificando suas funcionalidades, interconexões e comportamentos. Uma das principais vantagens da linguagem é sua capacidade de simulação, o que permite a verificação do funcionamento do sistema antes mesmo de sua implementação na placa.

2.3 DISPLAY BCD 7 SEGMENTOS:

O display BCD de 7 segmentos é um dispositivo que exibe números decimais e é muito utilizado para diversas aplicações. Cada dígito é representado por sete segmentos separados que podem ser ligados ou desligados, formando diferentes padrões que representam números de 0 a 9. Cada um desses sete segmentos representa uma parte específica do dígito e recebe uma letra de referência, podendo ser "a", "b", "c", "d", "e", "f" e "g" conforme mostra a imagem abaixo:



Sendo assim, para que o display exiba um valor numérico, o valor decimal precisa ser convertido para sua forma binária codificada em decimal (BCD), onde cada dígito decimal será representado por quatro bits binários. Em seguida, esses valores BCD são aplicados aos segmentos correspondentes do dispositivo, acendendo ou apagando os segmentos conforme necessário para exibir o número desejado.



2.4 TEMPORIZADOR:

Um temporizador é um dispositivo eletrônico utilizado para medir intervalos de tempo. Ele opera através da contagem de pulsos de clock, permitindo com que se torne possível determinar o tempo passado entre os chamados “eventos”.

O principal propósito desse dispositivo é controlar o início (ou término) de determinadas ações em sistemas automatizados. Quando configurado para um intervalo específico, o temporizador irá contar os ciclos de clock até atingir esse valor de referência, e então emitirá um sinal de saída que poderá ser utilizado para acionar outros processos. Em VHDL é possível implementar esse dispositivo descrevendo circuitos digitais a nível de hardware, bastando configurá-los especificando os parâmetros desejados (como frequência de clock e período de amostragem).

2.5 DEBOUNCER:

Um debouncer é um circuito (ou técnica) utilizado com a finalidade de filtrar e estabilizar os sinais de entrada que podem sofrer *bouncing* (ou rebote, como é chamado em português). O rebote ocorre no momento em que um botão é pressionado ou solto, resultando em transições rápidas entre 0 e 1 no estado lógico antes de se estabilizar em um estado final.

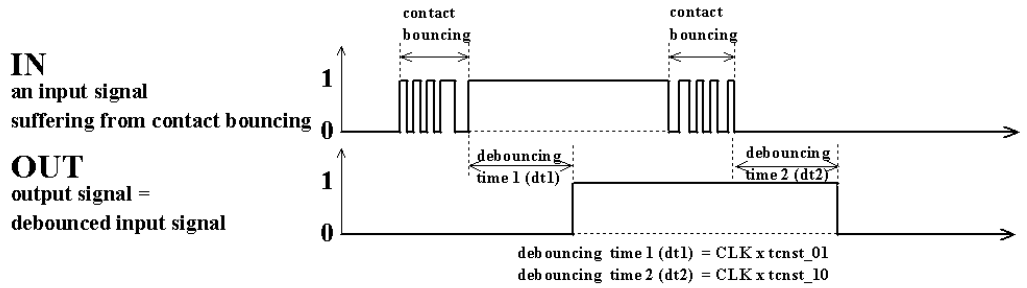


Figure 5. Timing diagram of each channel of the generic independent eight bit I/O contact debouncer.

Seu funcionamento ocorre da seguinte maneira: quando ocorre alteração no nível lógico da entrada (por exemplo, indo de 0 para 1), o debouncer aguarda um curto período de tempo a fim de permitir que qualquer oscilação no sinal se dissipe. Durante esse intervalo, o debouncer irá ignorar qualquer mudança que ocorrer na entrada, garantindo que apenas a primeira alteração seja registrada. Após esse intervalo, caso a entrada permaneça estável no seu novo estado, o debouncer irá considerar essa transição como válida e irá refleti-la na saída.

3 METODOLOGIA:

O presente capítulo visa explicar como foram realizados os dois projetos do terceiro laboratório da disciplina, onde em todos eles foram desenvolvidos códigos em VHDL para implementação e teste.

3.1 TEMPORIZADOR:

No projeto do temporizador, foram utilizadas as entradas *enable* e *reset* (STD_LOGIC) para habilitar e reiniciar a contagem. Além disso, uma entrada de clock (também STD_LOGIC) também foi necessária a fim de atender o requisito de contagem a cada meio segundo. Por fim, a saída *VETOR* – um vetor de 4 bits que representa um número de 0 a 9 em binário – foi utilizada para que *DISP* pudesse representar o dígito no display de 7 segmentos.

3.1.1 Códigos em VHDL:

Para o projeto, o código em VHDL abaixo foi desenvolvido:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4
5  ----------------------------------------------------------------------------------
6
7  entity TEMPORIZADOR is
8
9  generic (freq: integer := 50000; periodo: integer := 500); --50MHZ e 0.5s
10
11     Port (ena : in  STD_LOGIC; -- enable
12           clk : in  STD_LOGIC; -- clock
13           rst : in  STD_LOGIC; -- reset
14           anodo: out STD_LOGIC_VECTOR(3 downto 0); -- anodo
15           DISP : out STD_LOGIC_VECTOR(6 downto 0)); --display
16
17 end TEMPORIZADOR;
18
19 ----------------------------------------------------------------------------------
20
21 architecture Behavioral of TEMPORIZADOR is
22
23    signal VETOR : STD_LOGIC_VECTOR (3 downto 0);
24    constant maximo: integer := freq*periodo; -- calcula o tempo
25
26 begin
27
28    anodo <= "1110"; --liga apenas o display mais a direita
29
```

```
30    process (clk, rst)
31
32       variable cont: integer range 0 to maximo:= 0; --conta meio segundo
33       variable cont9: integer range 0 to 9 := 0; --conta 0 até 9
34
35       begin
36       if (rst = '1') then --garante que tudo esteja zerado caso rst tenha valor logico 1
37          cont9 := 0; --zera auxiliar
38          cont := 0; --zera o contador
39
40       elsif (clk'event and clk = '1') then --
41          if (ena = '1') then --e se enable for 1
42             cont := cont + 1; --conta quantas bordas de subida teve até o maximo
43             if (cont = maximo) then
44                cont9 := cont9 +1;
45                cont := 0;
46             if (cont9 = 10) then --QUANDO CHEGA EM 10 ZERA
47                cont9 := 0;
48             end if;
49          end if;
50       end if;
51    end if;
52
53       VETOR <= CONV_STD_LOGIC_VECTOR(cont9, 4);
```

```
54
55       case VETOR is
56       when "0000" => DISP <= "0000001";
57       when "0001" => DISP <= "1001111";
58       when "0010" => DISP <= "0010010";
59       when "0011" => DISP <= "0000110";
60       when "0100" => DISP <= "1001100";
61       when "0101" => DISP <= "0100100";
62       when "0110" => DISP <= "0100000";
63       when "0111" => DISP <= "0001111";
64       when "1000" => DISP <= "0000000";
65       when "1001" => DISP <= "0000100";
66       when others => DISP <= "1111111";
67       end case;
68
69    end process;
70
71 end Behavioral;
```

Para o seu funcionamento, são definidos dois parâmetros genéricos: *freq*, que especifica a frequência do clock em MHz; e *periodo*, que determina a duração da janela em segundos. A constante *maximo* é calculada multiplicando a frequência do clock pela duração da janela. Essa constante determina quantos pulsos de clock são necessários para abranger o período de tempo definido pela janela. Além disso, o código também introduz três sinais: *VETOR*, para armazenar um contador de 4 bits; *cont9*, para contar de 0 a 9 e selecionar o valor a ser exibido no display de 7 segmentos; e *cont*, que conta os pulsos de clock.

O processo em questão utiliza dois contadores: *cont* e *cont9*, onde *cont9* conta de 0 a 9 e é utilizado para selecionar o valor a ser exibido no display de 7 segmentos, enquanto *cont* conta os pulsos de clock e determina quando o temporizador deve ser incrementado.

A lógica de controle dentro do algoritmo verifica se o sinal de habilitação *ena* está ativo ('1'). Se estiver, o contador *cont* é incrementado a cada pulso de clock. Quando *cont* atinge o valor máximo (*maximo*), *cont9* é incrementado e *cont* é resetado.

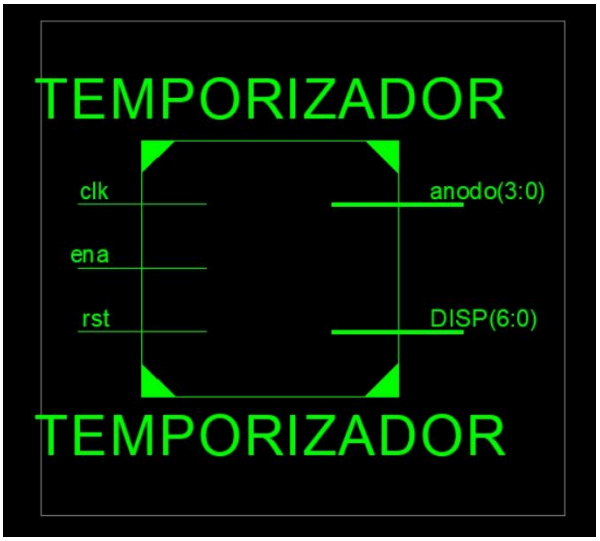
O código inclui uma estrutura *case* que associa os valores do temporizador para exibir números de 0 a 9 em um display de 7 segmentos. O valor correspondente aos segmentos do display de 7 segmentos é então atribuído ao sinal de saída *DISP*.

Esse processo continua a ser executado, atualizando o display de 7 segmentos conforme o valor do temporizador, e repete o ciclo.

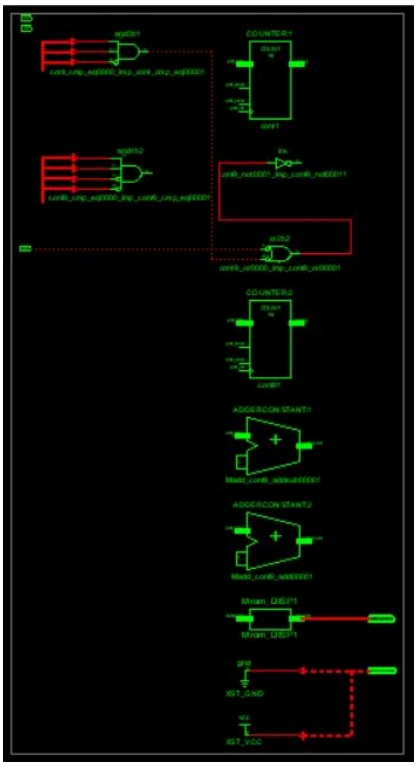
3.1.2 Esquema de pinos e esquemáticos:

```
1  NET "ena" LOC = "G18";
2  NET "rst" LOC = "H18";
3  NET "anodo(0)" LOC = "F17";
4  NET "anodo(1)" LOC = "H17";
5  NET "anodo(2)" LOC = "C18";
6  NET "anodo(3)" LOC = "F15";
7  NET "DISP(6)" LOC = "L18";
8  NET "DISP(5)" LOC = "F18";
9  NET "DISP(4)" LOC = "D17";
10 NET "DISP(3)" LOC = "D16";
11 NET "DISP(2)" LOC = "G14";
12 NET "DISP(1)" LOC = "J17";
13 NET "DISP(0)" LOC = "H14";
14 NET "clk" LOC = "B8";
```

O esquemático RTL (Register Transfer Logic) abaixo representa o comportamento do circuito digital a nível de registro:



Já o esquemático RTL Signals abaixo demonstra a disposição física dos componentes do circuito integrado em um nível mais detalhado do que os esquemáticos de nível lógico, como o RTL apresentado na figura acima.



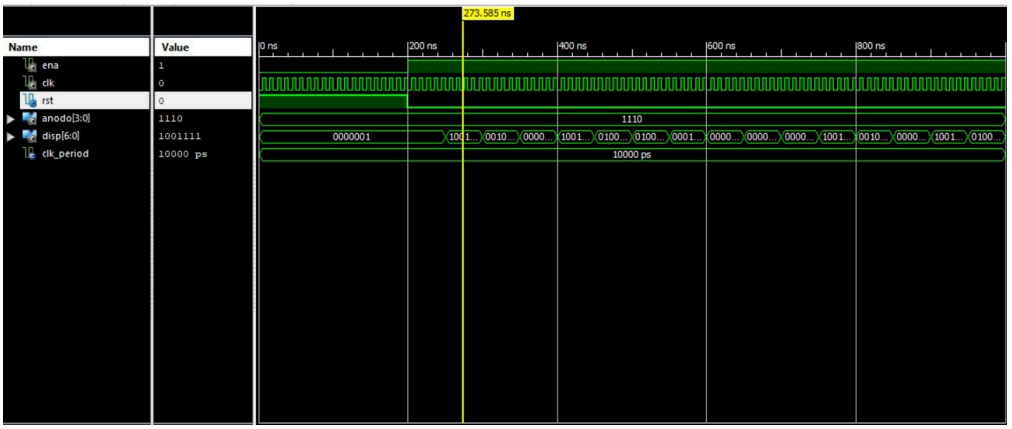
3.1.3 Teste:

Após o desenvolvimento do código e atribuição dos pinos, um *test bench* foi criado a fim de validar o comportamento do hardware descrito em VHDL, para que fosse possível simular seu funcionamento antes mesmo da implementação física na placa.

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY TEMPORIZADOR_TB IS
END TEMPORIZADOR_TB;

ARCHITECTURE behavior OF TEMPORIZADOR_TB IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT TEMPORIZADOR
    PORT(
         ena : IN  std_logic;
         clk : IN  std_logic;
         rst : IN  std_logic;
         anodo : OUT  std_logic_vector(3 downto 0);
         DISP : OUT  std_logic_vector(6 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal ena : std_logic := '0';
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';

   --Outputs
   signal anodo : std_logic_vector(3 downto 0);
   signal DISP : std_logic_vector(6 downto 0);

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: TEMPORIZADOR PORT MAP (
          ena => ena,
          clk => clk,
          rst => rst,
          anodo => anodo,
          DISP => DISP
        );

   -- 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
```

```
60      -- Stimulus process
61       stim_proc: process
62      begin
63         rst <= '1';
64         ena <= '0';
65         wait for 200 ns;
66         rst <= '0';
67         ena <= '1';
68         wait for clk_period*100;
69
70         wait;
71      end process;
72
73  END;
74
```

Quando diferentes combinações de entradas são inseridas, o resultado é apresentado na figura abaixo:



Na figura acima, é apresentada uma estrutura que mapeia os valores para os padrões de segmentos apropriados, permitindo a exibição dos números de 0 a 9 em um display de 7 segmentos. Dessa maneira, o valor mapeado para os segmentos do display de 7 segmentos é atribuído ao sinal de saída. O processo continua em execução, atualizando o display de 7 segmentos.

Com base no gráfico da simulação, podemos concluir que o temporizador construído em VHDL funciona adequadamente, atendendo aos requisitos estabelecidos.

3.1.4 Utilização do dispositivo e considerações:

Abaixo é possível observar o resumo dos recursos lógicos utilizados:

TEMPORIZADOR Project Status (04/19/2024 - 17:41:17)			
Project File:	TEMPORIZADOR.xise	**Parser Errors:**	No Errors
Module Name:	TEMPORIZADOR	**Implementation State:**	Placed and Routed
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	7	9,312	1%	
Number of 4 input LUTs	14	9,312	1%	
Number of occupied Slices	7	4,656	1%	
Number of Slices containing only related logic	7	7	100%	
Number of Slices containing unrelated logic	0	7	0%	
Total Number of 4 input LUTs	14	9,312	1%	
Number of bonded IOBs	14	232	6%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	4.29			

3.2 CONTADOR 0-9:

No segundo projeto, foi desenvolvido um código para um contador de 0 a 9 que incrementa ou decrementa cada vez que o botão da placa é pressionado.

3.2.1 Códigos em VHDL:

Para o projeto, o código em VHDL abaixo foi desenvolvido:

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

----------------------------------------------------------------------
entity contador_module is

      generic (freq: INTEGER := 50000; periodo: integer := 500); --50MHz
e 500ms

    Port ( ena : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           push_b : in  STD_LOGIC;
           up_down : in  STD_LOGIC;
           dig : out  STD_LOGIC_VECTOR (6 downto 0);
```

```
                  anodo : out STD_LOGIC_VECTOR (3 downto 0));

end contador_module;

----------------------------------------------------------------------------

architecture Behavioral of contador_module is

      signal aux_cont: STD_LOGIC_VECTOR (3 downto 0);
      signal deb: STD_LOGIC := '0';

begin
      anodo <= "1110";
----------------------------------------------------------------------------
process(clk, rst)
      variable aux_clk: integer range 0 to (freq * periodo) := 0;
begin
      if (rst = '1') then
            aux_clk := 0;

      elsif (clk'event and clk = '1') then
            if (push_b /= deb) then
                  aux_clk := aux_clk + 1;
                  if (aux_clk = freq * periodo) then
                        deb <= push_b;
                        aux_clk := 0;
                  end if;
            else
                  aux_clk := 0;
            end if;
      end if;
end process;
----------------------------------------------------------------------------
process(rst, deb, aux_cont)
      variable cont: integer range 0 to 10 := 0; -- variavel do contador
begin
            if (rst = '1') then
                  cont := 0;
```

```
            elsif (deb'event and deb = '1') then
                  if(ena = '1') then
                        if (up_down = '0') then
                              if (cont = 0) then
                                    cont := 9;
                              else
                              cont := cont - 1;
                              end if;
                        else
                              if(cont = 9) then
                                    cont := 0;
                              else
                              cont := cont + 1;
                              end if;
                        end if;
                  end if;
      end if;

      aux_cont <= CONV_STD_LOGIC_VECTOR(cont, 4);

      case aux_cont is
            when "0000" => dig <= "0000001";
            when "0001" => dig <= "1001111";
            when "0010" => dig <= "0010010";
            when "0011" => dig <= "0000110";
            when "0100" => dig <= "1001100";
            when "0101" => dig <= "0100100";
            when "0110" => dig <= "0100000";
            when "0111" => dig <= "0001111";
            when "1000" => dig <= "0000000";
            when "1001" => dig <= "0000100";
            when others => dig <= "1111111";
      end case;
end process;
-------------------------------------------------------------------------
end Behavioral;
```

Primeiramente, é declarada a frequência e a janela de funcionamento do clock. Os valores utilizados para implementação na placa foram de 50MHz e 10ms, respectivamente. Em seguida, foram declarados dois sinais: *aux_cont*, como vetor auxiliar de *cont* com 4 bits; e *deb*, variável representando debouncer. O display mais à direita foi utilizado para mostrar os dígitos, então *anodo* recebeu a configuração 1110.

Para que fosse possível obter um sistema contando de 0 a 9 que incrementa ou decrementa a cada vez que o botão é pressionado, foram utilizados dois processos: o primeiro (com *clk* e *rst* na lista de sensibilidade) é responsável pela contagem de ciclos do clock a fim de detectar se houve alguma mudança no estado do botão, isto é, se ele foi pressionado ou solto; enquanto o segundo processo (com *rst*, *deb* e *aux_cont* na lista de sensibilidade) atua como o contador em si.

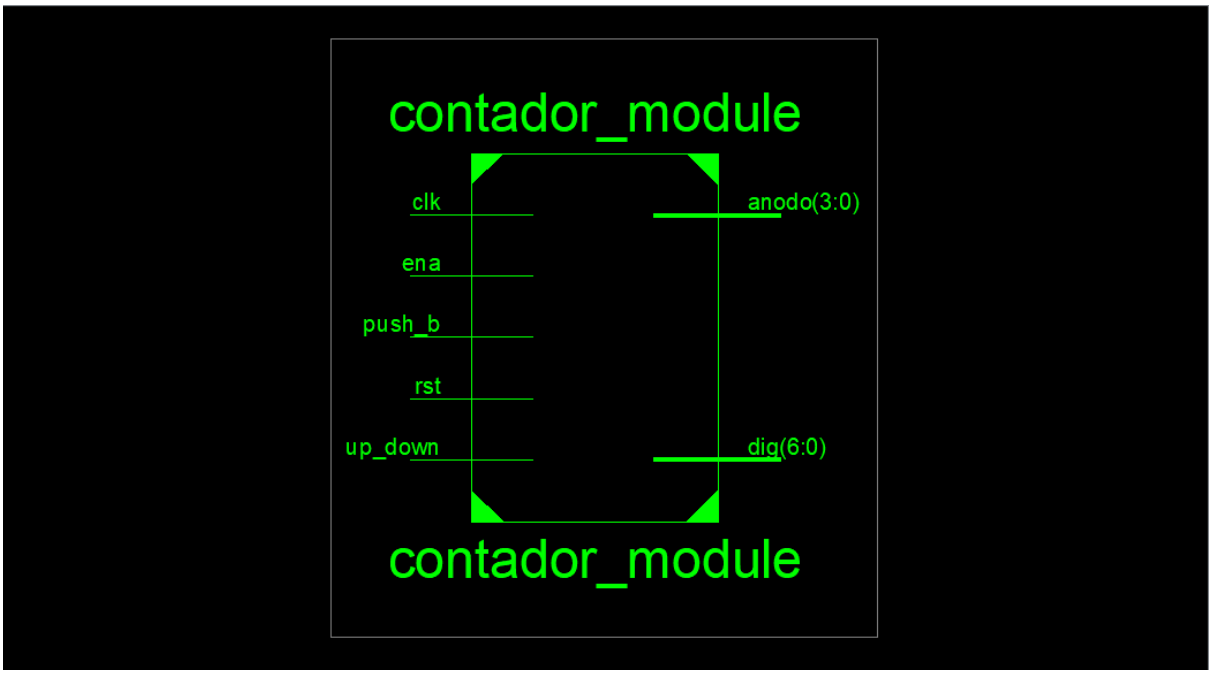
Em suma, se houver mudança no estado do clock e ele assumir nível lógico alto, e se o nível lógico do botão for diferente do estado de debouncing, então a variável auxiliar de clock *aux_clk* é incrementada. Quando *aux_clk* atinge seu valor máximo, isto é, o valor da frequência de clock multiplicada pela janela de funcionamento, então a variável *deb* recebe o valor correspondente ao nível lógico do botão e *aux_clk* assume valor zerado. Caso o estado do botão seja o mesmo do estado de debouncing, *aux_clk* assume valor zerado. Quando uma borda de subida ocorre no sinal de debouncing e a variável *ena* tem valor lógico 1, o sinal *up_down* é verificado a fim de determinar se cont será incrementado ou decrementado. Caso *up_down* seja igual a 0, *cont* é decrementado; caso contrário, *cont* é incrementado. O vetor auxiliar *aux_cont* recebe então o valor que foi armazenado na variável *cont* e depois convertido em um vetor de 4 bits utilizando *CONV_STD_LOGIC_VECTOR*. Ao final do processo, o vetor *dig* de 7 bits recebe a sequência correspondente a cada número de 0 a 9 armazenado em *aux_cont*, mostrando-o no display.

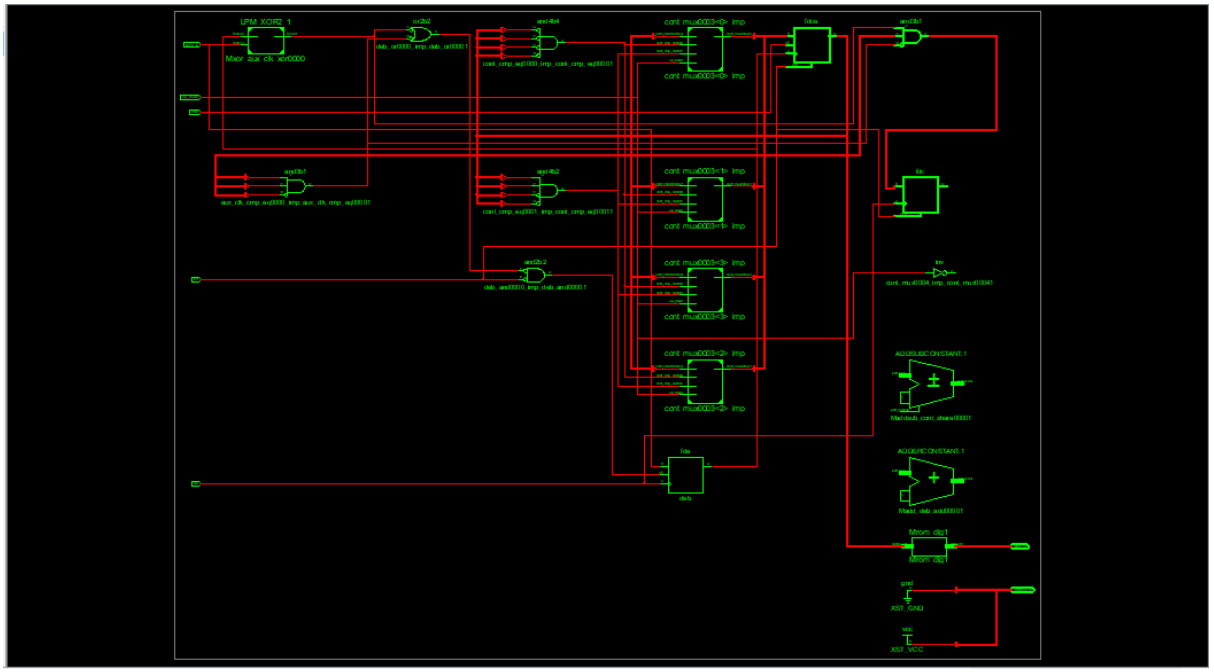
3.2.2 Esquema de pinos e esquemáticos:

A figura abaixo demonstra o esquema de pinos na placa, onde cada porta de *dig* representa um dos segmentos de cada display. Para que apenas o display mais à direita fique aceso, o bit menos significativo (com nível lógico zero) foi atribuído à porta F17.

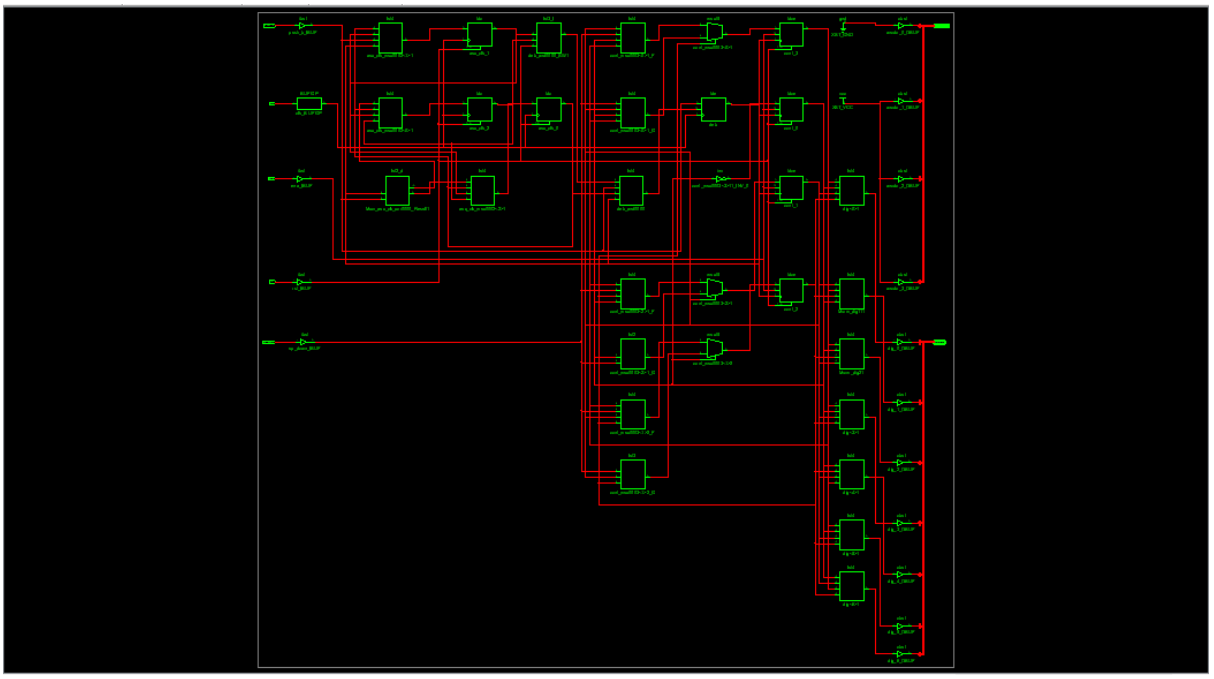
```
1   NET "clk" LOC = "B8";
2   NET "ena" LOC = "G18";
3   NET "rst" LOC = "H18";
4   NET "up_down" LOC = "K18";
5   NET "push_b" LOC = "H13";
6
7   NET "anodo(0)" LOC = "F17";
8   NET "anodo(1)" LOC = "H17";
9   NET "anodo(2)" LOC = "C18";
10  NET "anodo(3)" LOC = "F15";
11
12  NET "dig(6)" LOC = "L18";
13  NET "dig(5)" LOC = "F18";
14  NET "dig(4)" LOC = "D17";
15  NET "dig(3)" LOC = "D16";
16  NET "dig(2)" LOC = "G14";
17  NET "dig(1)" LOC = "J17";
18  NET "dig(0)" LOC = "H14";
```

O esquemático RTL (Register Transfer Logic) abaixo representa o comportamento do circuito digital a nível de registro:





Já o esquemático de tecnologia abaixo demonstra a disposição física dos componentes do circuito integrado em um nível mais detalhado do que os esquemáticos de nível lógico, como o RTL apresentado na figura acima.



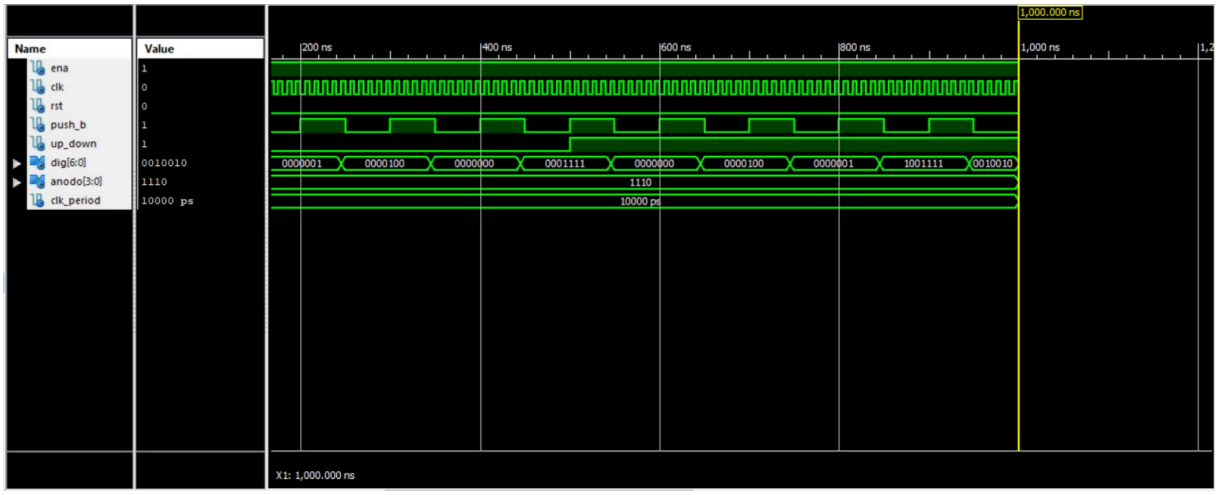
3.2.3 Teste:

Após o desenvolvimento do código e atribuição dos pinos, um *test bench* foi criado a fim de validar o comportamento do hardware descrito em VHDL, para que fosse possível simular seu funcionamento antes mesmo da implementação física na placa.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY contador_tb IS
5  END contador_tb;
6
7  ARCHITECTURE behavior OF contador_tb IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10
11     COMPONENT contador_module
12     PORT(
13          ena : IN  std_logic;
14          clk : IN  std_logic;
15          rst : IN  std_logic;
16          push_b : IN  std_logic;
17          up_down : IN  std_logic;
18          dig : OUT  std_logic_vector(6 downto 0);
19          anodo : OUT  std_logic_vector(3 downto 0)
20         );
21     END COMPONENT;
22
23
24    --Inputs
25    signal ena : std_logic := '0';
26    signal clk : std_logic := '0';
27    signal rst : std_logic := '0';
28    signal push_b : std_logic := '0';
29    signal up_down : std_logic := '0';
30
31    --Outputs
```

```
31    --Outputs
32    signal dig : std_logic_vector(6 downto 0);
33    signal anodo : std_logic_vector(3 downto 0);
34
35    -- Clock period definitions
36    constant clk_period : time := 10 ns;
37
38  BEGIN
39
40    -- Instantiate the Unit Under Test (UUT)
41    uut: contador_module PORT MAP (
42           ena => ena,
43           clk => clk,
44           rst => rst,
45           push_b => push_b,
46           up_down => up_down,
47           dig => dig,
48           anodo => anodo
49         );
50
51    -- Clock process definitions
52    clk_process :process
53    begin
54       clk <= '0';
55       wait for clk_period/2;
56       clk <= '1';
57       wait for clk_period/2;
58    end process;
59
60   btt :process
61     begin
```

```
61     begin
62         push_b <= '1';
63        wait for 50  ns;
64         push_b <= '0';
65        wait for 50 ns;
66     end process;
67
68     stim_proc: process
69     begin
70        rst <= '0';
71        ena <= '1' after 50 ns;
72        up_down <= '1' after 500 ns;
73        wait for 50 ns;
74        rst <= '1';
75        wait for 100 ns;
76        rst <= '0';
77        wait for 300 ns;
78        rst <= '0';
79        wait for 400 ns;
80        rst <= '0';
81        wait for 400 ns;
82        wait;
83     end process;
84
85  END;
```



Através do resultado da simulação realizada, é possível notar, por exemplo, que quando enable está ativado, reset zerado e *up_down* em valor lógico alto, a cada vez que o botão é pressionado, *dig* (representando o dígito no display) cresce de 0 a 9, em ordem crescente.

3.2.4 Utilização do dispositivo e considerações:

Conforme é possível observar na imagem abaixo, apenas 1% de LUTs (estrutura de dados) de 4 inputs foi utilizado.

contador_module Project Status (04/19/2024 - 17:32:12)			
Project File:	lab3_contador.xise	**Parser Errors:**	No Errors
Module Name:	contador_module	**Implementation State:**	Programming File Generated
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	1 Warning (1 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	24	9,312	1%	
Number of 4 input LUTs	39	9,312	1%	
Number of occupied Slices	31	4,656	1%	
Number of Slices containing only related logic	31	31	100%	
Number of Slices containing unrelated logic	0	31	0%	
Total Number of 4 input LUTs	57	9,312	1%	
Number used as logic	39			
Number used as a route-thru	18			
Number of bonded IOBs	16	232	6%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.82			

Performance Summary				[-]
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report	
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report	
Timing Constraints:	All Constraints Met			

Detailed Reports						[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**	
Synthesis Report	Current	sex 19. abr 17:31:40 2024	0	0	2 Infos (1 new)	
Translation Report	Current	sex 19. abr 17:31:44 2024	0	0	0	
Map Report	Current	sex 19. abr 17:31:49 2024	0	0	2 Infos (0 new)	
Place and Route Report	Current	sex 19. abr 17:31:57 2024	0	1 Warning (1 new)	2 Infos (1 new)	
Power Report						
Post-PAR Static Timing Report	Current	sex 19. abr 17:31:59 2024	0	0	0	
Bitgen Report	Current	sex 19. abr 17:32:09 2024	0	0	0	

Secondary Reports			[-]
Report Name	**Status**	**Generated**	
ISIM Simulator Log	Out of Date	sex 19. abr 17:38:46 2024	
WebTalk Report	Out of Date	sex 19. abr 17:32:09 2024	
WebTalk Log File	Out of Date	sex 19. abr 17:32:11 2024	

4 RESULTADOS E CONCLUSÃO:

Cada projeto abordou diferentes sistemas cujas lógicas se complementam entre si, passando pela criação de um temporizador até o desenvolvimento de um contador de 0 a 9 que incrementa ou decrementa a cada vez que o botão da placa é pressionado. Tanto no primeiro projeto quanto no segundo, a linguagem VHDL foi utilizada para escrever o test bench, permitindo com que fosse possível realizar a correção de erros e validação do funcionamento adequado dos sistemas antes de implementá-los na placa.

Dessa maneira, os resultados obtidos nos projetos do laboratório demonstram a aplicação prática dos conceitos teóricos estudados, bem como também a importância da simulação dos sistemas antes da implementação desses.

5 REFERÊNCIA:

França, S. B. L. (Prof. Dr.). (2024). Aulas de Microeletrônica 1, ministradas pela Prof. Dr. Sibilla Batista da Luz França. UFPR.