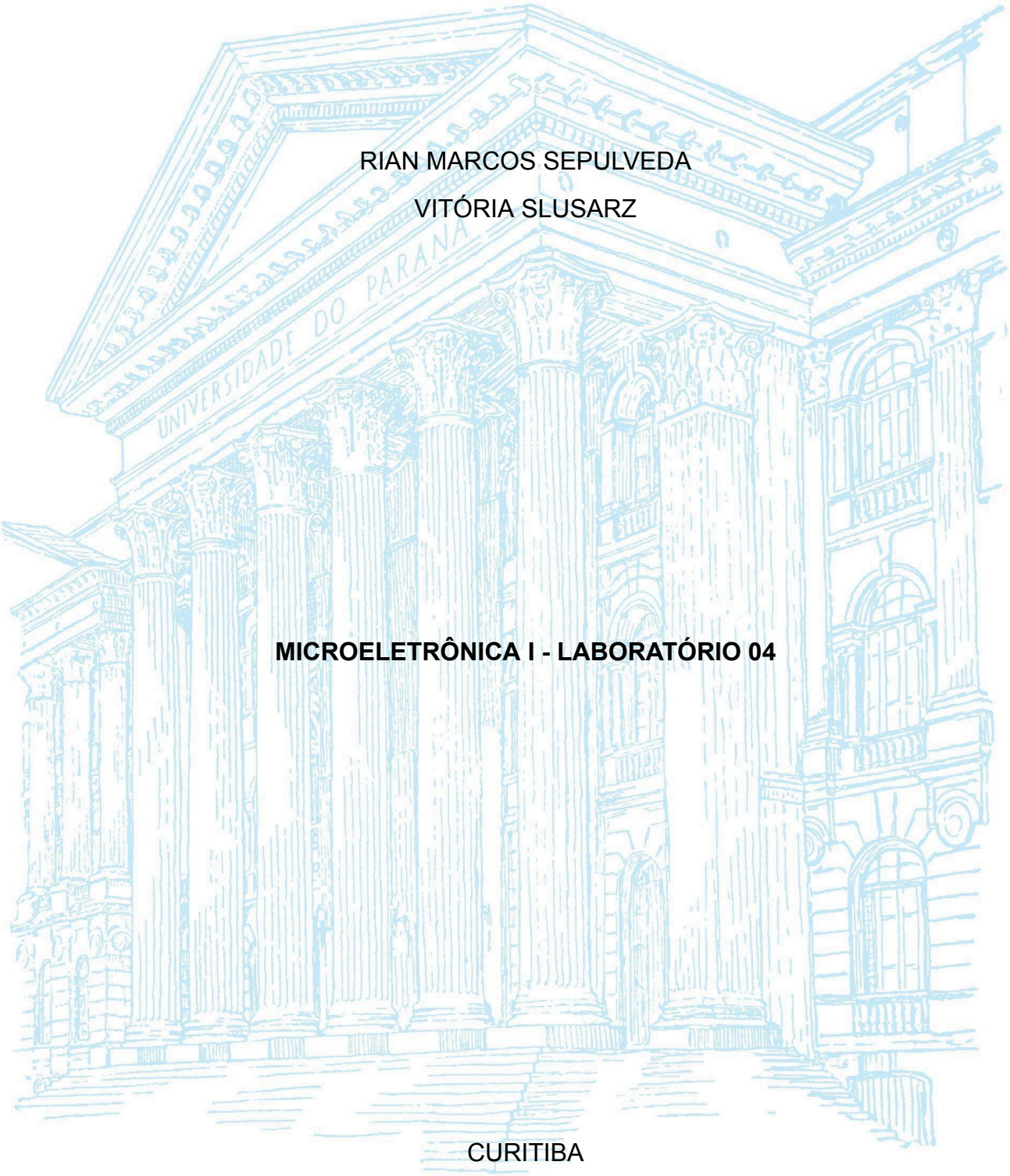
UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
SETOR DE TECNOLOGIA



RIAN MARCOS SEPULVEDA
VITÓRIA SLUSARZ

MICROELETRÔNICA I - LABORATÓRIO 04

CURITIBA
2024

RIAN MARCOS SEPULVEDA
VITÓRIA SLUSARZ

MICROELETRÔNICA I - LABORATÓRIO 04

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 o experimento realizado no quarto laboratório da disciplina de Microeletrônica I, onde foi desenvolvido um cronômetro que realiza a contagem das horas, minutos e segundos. 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

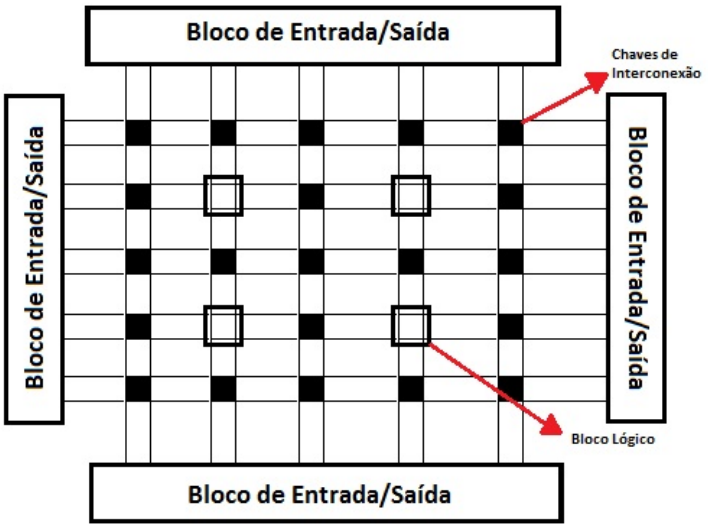
Neste relatório, são apresentados os resultados de um laboratório prático dedicado à implementação de um cronômetro de horas, minutos e segundos utilizando linguagem de descrição de hardware VHDL e a placa FPGA XC3S500E. Além dos resultados, são detalhadas todas as etapas do projeto, incluindo simulação e teste do circuito.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo visa oferecer uma base conceitual para compreender os princípios relacionados aos temas abordados no experimento 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 como base para elaboração do projeto.

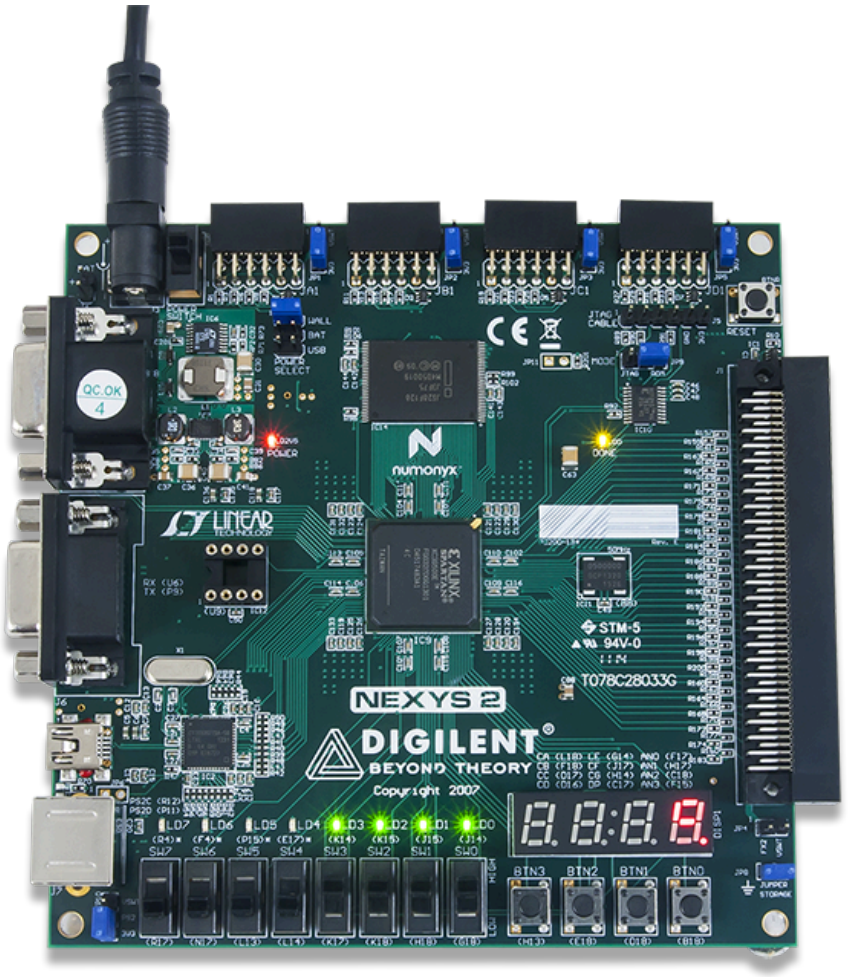
2.1 FPGA - XC3S500E:

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.

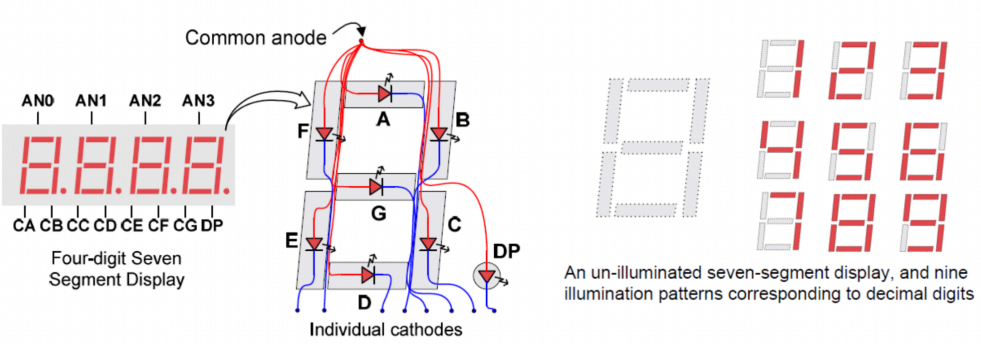
2.2 VHDL:

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

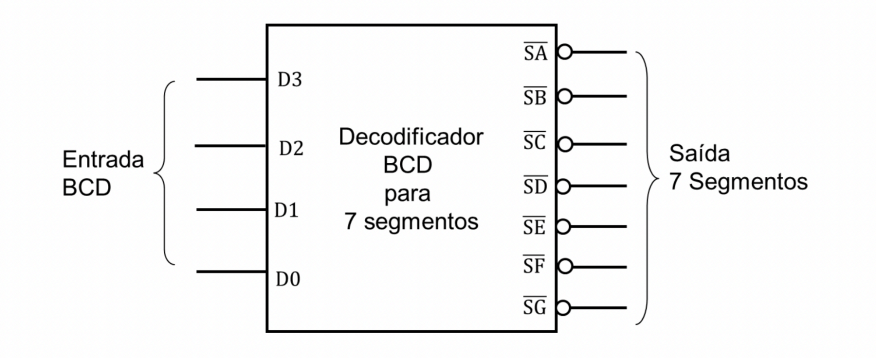
pequenos componentes lógicos até sistemas mais completos, especificando suas funcionalidades, interconexões e comportamentos. Uma das principais vantagens da linguagem é sua capacidade de simulação, o que permite a verificação do funcionamento do sistema antes mesmo de sua implementação na placa.

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:

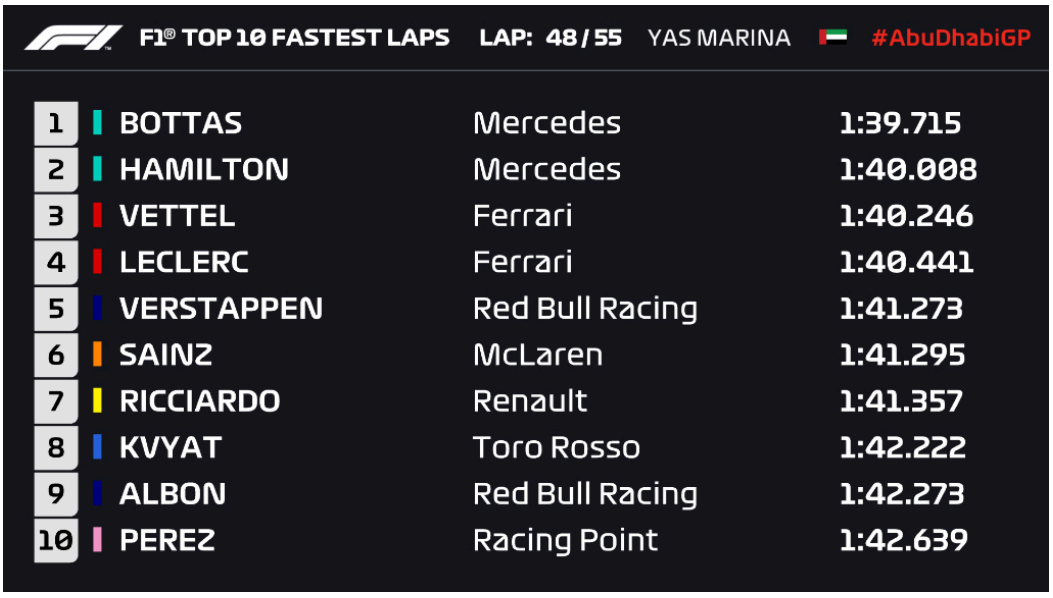


Sendo assim, para que o display exiba um valor numérico, o valor decimal precisa ser convertido para sua forma binária codificada em decimal (BCD), onde cada dígito decimal será representado por quatro bits binários. Em seguida, esses valores BCD são aplicados aos segmentos correspondentes do dispositivo,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



2.4 CRONÔMETRO:

Um cronômetro nada mais é que um dispositivo utilizado para a medição do tempo, permitindo a contagem de intervalos temporais. O dispositivo funciona através da marcação do tempo decorrido desde um ponto de partida (início) até o momento atual. A operação de um cronômetro tem como base um circuito interno que controla o início, pausa e reset, e normalmente os valores são cronometrados em horas, minutos e segundos, mas para esportes como Fórmula 1, por exemplo, são utilizadas também as frações de segundo. Na imagem abaixo é possível observar que, para o ranking das dez voltas mais rápidas no Grande Prêmio de Abu Dhabi, o centésimo de segundo teve papel essencial para ordenar dois ou mais pilotos cuja duração da volta foi de, por exemplo, 1 minuto e 40 segundos.



3 METODOLOGIA:

O presente capítulo visa explicar como foi realizado o projeto do quarto laboratório da disciplina, desenvolvido em VHDL para implementação e teste.

3.1 CRONÔMETRO:

Para o projeto do cronômetro, foram criados dois processos no documento principal do circuito, responsáveis pela lógica do relógio e do funcionamento dos displays, uma vez que com o kit Nexus 2 os quatro displays são interligados entre si. Esse detalhe fez com que fosse necessária a criação de um processo destinado unicamente para alternar a exibição dos displays de maneira que fosse imperceptível para o olho humano notar. Além disso, foi criado um documento auxiliar responsável pela conversão de inteiros em vetores de 7 bits.

3.1.1 Códigos em VHDL:

Abaixo são descritos os trechos de código VHDL que compõem o programa principal e o auxiliar.

3.1.1.1 Código principal “relogio_module”

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

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity relogio_module is

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

    Port ( rst : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           pausa : in  STD_LOGIC;
           conf : in  STD_LOGIC;
           dig_x: out STD_LOGIC_VECTOR (6 downto 0);
          anodo : out STD_LOGIC_VECTOR (3 downto 0));

end relogio_module;

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

architecture Behavioral of relogio_module is

   constant max: integer := 50000000; --50 milhoes
   constant max_clk: integer := 8000;

   component aux_relogio_module is
      Port ( d_x: in integer;
             dig_x_internal: out STD_LOGIC_VECTOR (6 downto 0)); -- display inicializa
apagado
   end component;

   signal d_x : integer; -- Nova variável interna para armazenar o valor de d_x
   signal d_x_internal : integer; -- Nova variável interna para armazenar o valor de
d_x
   signal anodo_aux: STD_LOGIC_VECTOR(3 downto 0) := "1110";

begin

d_to_dig: aux_relogio_module PORT MAP (d_x=>d_x, dig_x_internal=>dig_x);
----------------------------------------------------------------------------


   --processo referente ao display
   process (clk)

      variable aux_clk: integer := 0;

      begin

      if(clk'event and clk = '1') then
         aux_clk := aux_clk + 1;
         if(aux_clk = max_clk) then
            aux_clk := 0;
            if(anodo_aux = "1110") then
               anodo <= "1101";
               anodo_aux <= "1101";
            elsif(anodo_aux  = "1101") then
               anodo <= "1011";
```

```
56                  anodo_aux <= "1011";
57               elsif(anodo_aux  = "1011") then
58                  anodo <= "0111";
59                  anodo_aux <= "0111";
60               elsif(anodo_aux  = "0111") then
61                  anodo <= "1110";
62                  anodo_aux <= "1110";
63               end if;
64            end if;
65         end if;
66
67      end process;
68
69   ----------------------------------------------------------------------------
70
71      --processo referente ao relógio
72      process (clk, rst)
73
74      --variaveis correspondentes aos displays
75      variable d_1: integer range 0 to 10 := 0;
76      variable d_2: integer range 0 to 10 := 0;
77      variable d_3: integer range 0 to 10 := 0;
78      variable d_4: integer range 0 to 10 := 0;
79      variable d_5: integer range 0 to 10 := 0;
80      variable d_6: integer range 0 to 10 := 0;
81
82      variable cont: integer range 0 to max := 0;
83
84      --signal aux_cont: STD_LOGIC_VECTOR (3 downto 0);
85
86         begin
87
88         --conf = '1', entao horas e minutos
89          if(conf = '1') then
90                  if(anodo_aux = "1110") then
91                     d_x <= d_3;
92                  elsif(anodo_aux = "1101") then
93                     d_x <= d_4;
94                  elsif(anodo_aux = "1011") then
95                     d_x <= d_5;
96                  elsif(anodo_aux = "0111") then
97                     d_x <= d_6;
98                  end if;
99
100         elsif(conf = '0') then --conf = '0', entao minutos e segundos
101                 if(anodo_aux = "1110") then
102                    d_x <= d_1;
103                 elsif(anodo_aux = "1101") then
104                    d_x <= d_2;
105                 elsif(anodo_aux = "1011") then
106                    d_x <= d_3;
107                 elsif(anodo_aux = "0111") then
108                    d_x <= d_4;
109                 end if;
110         end if;
111
112         if(rst = '1') then
```

```
113         d_1 := 0;
114         d_2 := 0;
115         d_3 := 0;
116         d_4 := 0;
117         d_5 := 0;
118         d_6 := 0;
119         cont := 0;
120 
121         elsif(clk'event and clk = '1') then
122            cont := cont + 1;
123            if(pausa = '0') then
124               if(cont = max) then
125                  d_1 := d_1 + 1;
126                  if(d_1 = 10) then
127                     d_1 := 0;
128                     d_2 := d_2 + 1;
129                     if(d_2 = 6) then
130                        d_1 := 0;
131                        d_2 := 0;
132                        d_3 := d_3 + 1;
133                        if(d_3 = 10) then
134                           d_1 := 0;
135                           d_2 := 0;
136                           d_3 := 0;
137                           d_4 := d_4 + 1;
138                           if(d_4 = 6) then
139                              d_1 := 0;
140                              d_2 := 0;
141                              d_3 := 0;
142                              d_4 := 0;
143                              d_5 := d_5 + 1;
144                              if(d_5 = 10) then
145                                 d_1 := 0;
146                                 d_2 := 0;
147                                 d_3 := 0;
148                                 d_4 := 0;
149                                 d_5 := 0;
150                                 d_6 := d_6 + 1;
151                                 if(d_6 = 6 and d_5 = 9) then
152                                    d_1 := 0;
153                                    d_2 := 0;
154                                    d_3 := 0;
155                                    d_4 := 0;
156                                    d_5 := 0;
157                                    d_6 := 0;
158                                 end if; --d_6
159                              end if; --d_5
160                           end if; --d_4
161                        end if; --d_3
162                     end if; --d_2
163                  end if; --d_1
164               end if; --cont
165            end if; --pausa
166         end if; --rst e elsif
167 
168      end process;
169 

170   ------------------------------------------------------------------------
171 
172   end Behavioral;
```

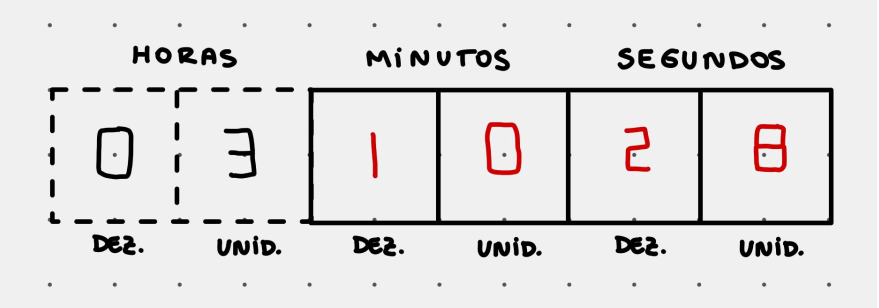
No trecho de código referente ao display, duas variáveis auxiliares tiveram que ser utilizadas, sendo elas *aux_clk* para o clock, e *anodo_aux* para o anodo. Dessa forma, toda vez que ocorre uma borda de subida no sinal de clock, o contador *aux_clk* é incrementado. Assim que *aux_clk* atinge seu valor máximo (nesse caso,

oito mil), o contador é resetado e o vetor *anodo* de 4 bits (responsável pela ativação dos displays) assume uma nova sequência de bits, bem como também seu auxiliar *anodo_aux*. Uma vez que não é possível comparar o estado atual de *anodo* com uma sequência de bits, a variável *anodo_aux* é utilizada para esse fim, na intenção de que *anodo* seja alterado e outro display passe a mostrar o dígito em questão.

O trecho de código onde *d_x* é alterado será explicado mais abaixo, juntamente com a lógica do componente *aux_relogio_module*. Já o trecho de código do processo do relógio é responsável por atualizar os valores referentes às dezenas e unidades das horas, minutos e segundos, onde cada variável declarada corresponde a cada uma dessas atribuições, sendo *d_1* responsável pela unidade dos segundos, e *d_6* responsável pela dezena da hora. Além disso, a variável *cont* também foi utilizada para a contar o número de ciclos de clock. Caso *rst* esteja em nível lógico alto, todas as variáveis são zeradas. Caso contrário, se ocorrer uma borda de subida do clock, o contador *cont* é incrementado. Se o sinal de pausa (*pausa*) estiver baixo, o relógio continua a contar normalmente. Quando o contador atinge o valor máximo (neste caso, cinquenta milhões), os contadores dos dígitos são atualizados de acordo com o tempo decorrido. Para as unidades de tempo, toda vez que *d_1*, *d_3* e *d_5* são iguais a 10, então eles próprios e os contadores anteriores a eles são zerados, enquanto o contador imediatamente posterior a cada um é incrementado em um. Para as dezenas, o mesmo processo ocorre quando *d_2*, *d_4* e *d_6* são iguais a 6. No projeto em questão, esses valores (10 e 6) foram utilizados devido ao fato das variáveis terem sido declaradas como *variable* ao invés de *signal*. Caso a comparação fosse com o número 9, por exemplo, o display "pularia" a exibição do dígito 9 e mostraria o número 0 no lugar.

Para o projeto, foi considerado o limite máximo de contagem do cronômetro em 59 horas, 59 minutos e 59 segundos. Dessa maneira, assim que o cronômetro atinge esse valor, todos os displays são zerados.

É importante destacar que todos os contadores dos dígitos continuam atuando simultaneamente, mesmo que não estejam sendo exibidos nos 4 displays, conforme demonstra a figura abaixo, onde a hora não é mostrada no display, mas continua sendo incrementada:



3.1.1.2 Código do componente “aux_relogio_module”:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity aux_relogio_module is
5
6      Port ( d_x : in  integer;
7        dig_x_internal : out  STD_LOGIC_VECTOR (6 downto 0) := "1111111");
8
9  end aux_relogio_module;
10
11 architecture Behavioral of aux_relogio_module is
12
13 begin
14
15       dig_x_internal <= "0000001" when d_x = 0 else
16       "1001111" when d_x = 1 else
17       "0010010" when d_x = 2 else
18       "0000110" when d_x = 3 else
19       "1001100" when d_x = 4 else
20       "0100100" when d_x = 5 else
21       "0100000" when d_x = 6 else
22       "0001111" when d_x = 7 else
23       "0000000" when d_x = 8 else
24       "0000100" when d_x = 9 else
25       "1111111";
26
27 end Behavioral;
```

O componente *aux_relogio_module* é declarado no início do programa e serve como auxiliar para o sistema de exibição do relógio, onde *d_x* é uma entrada genérica que recebe contadores de *d_1* a *d_6*, e *dig_x* uma saída de 7 bits que como padrão tem todos os segmentos do display apagados. Com base no sinal *conf*, é definido se os displays mostrarão as horas e os minutos (*conf* em nível alto), ou os minutos e os segundos (*conf* em nível baixo). Além disso, *d_x* assumirá o valor de algum contador de dígitos específico de acordo com o estado atual de *anodo*. Por exemplo, caso *conf* esteja em nível lógico baixo e o estado de *anodo* seja "1110",

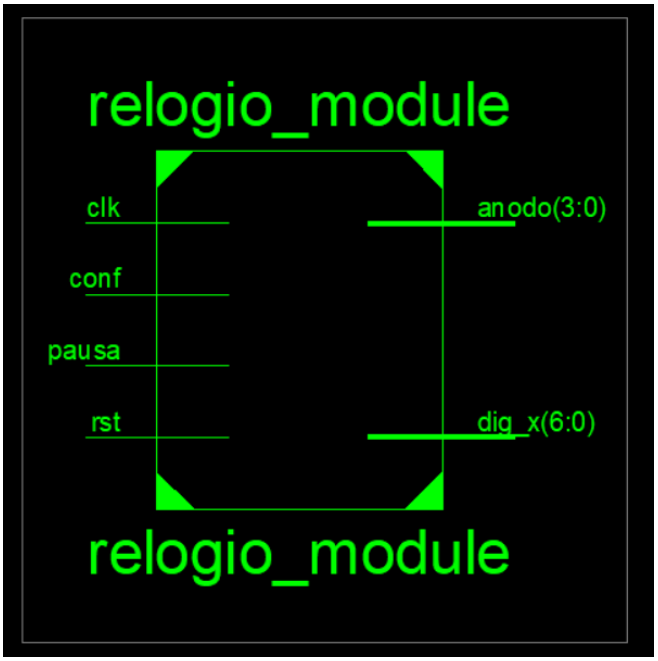
d_x irá assumir o inteiro armazenado em *d_1*, contador responsável pela unidade dos segundos.

3.1.2 Esquema de pinos e esquemáticos:

A figura abaixo representa o esquema de pinos utilizado:

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

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



3.1.3 Teste:

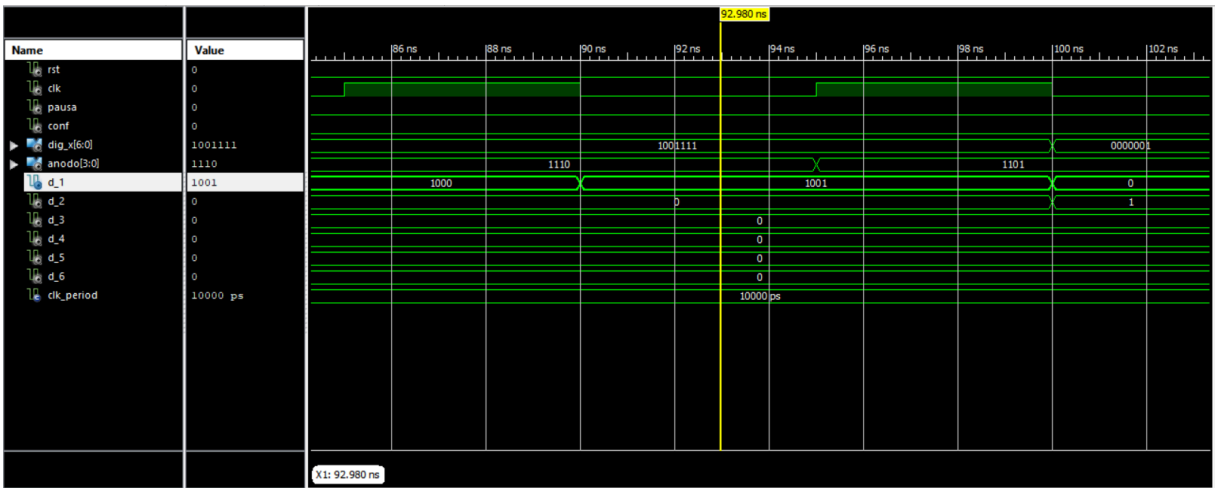
Após o desenvolvimento do código e atribuição dos pinos, um *test bench* foi criado a fim de validar o comportamento do hardware descrito em VHDL, para que fosse possível simular seu funcionamento antes mesmo da implementação física na placa.

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

ENTITY relogio_module_tb IS
END relogio_module_tb;

ARCHITECTURE behavior OF relogio_module_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT relogio_module
    PORT(
         rst : IN  std_logic;
         clk : IN  std_logic;
         pausa : IN  std_logic;
         conf : IN  std_logic;
         dig_x : OUT  std_logic_vector(6 downto 0);
         anodo : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

    --Inputs
    signal rst : std_logic := '0';
    signal clk : std_logic := '0';
    signal pausa : std_logic := '0';
    signal conf : std_logic := '0';

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

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

    -- Counter values for simulation
```

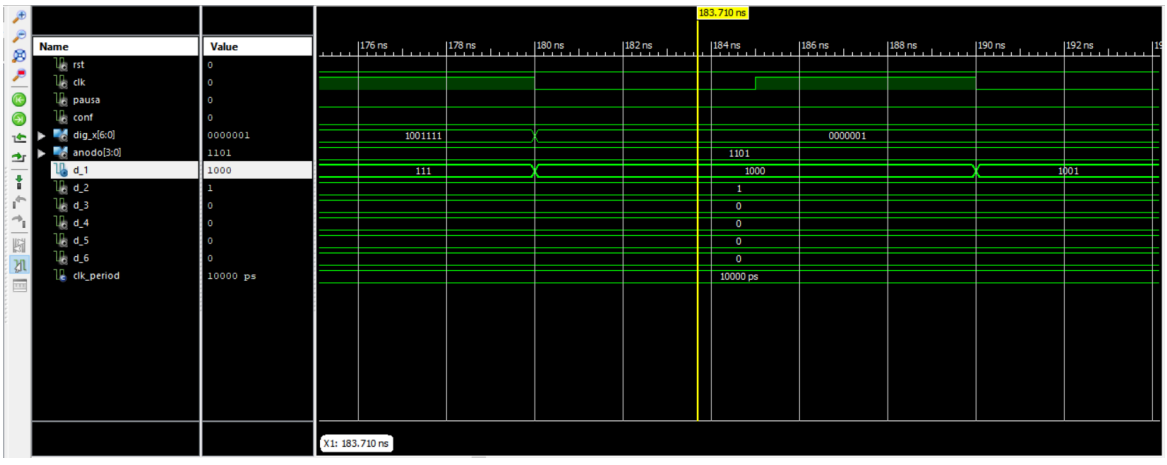
```vhdl
    -- Counter values for simulation
    signal d_1 : integer range 0 to 10 := 0;
    signal d_2 : integer range 0 to 10 := 0;
    signal d_3 : integer range 0 to 10 := 0;
    signal d_4 : integer range 0 to 10 := 0;
    signal d_5 : integer range 0 to 10 := 0;
    signal d_6 : integer range 0 to 10 := 0;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: relogio_module PORT MAP (
          rst => rst,
          clk => clk,
          pausa => pausa,
          conf => conf,
          dig_x => dig_x,
          anodo => anodo
        );

    -- 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

    -- Simulation of counter values
```

```
67      -- Simulation of counter values
68          for i in 0 to 599 loop
69              d_1 <= i mod 10;
70              d_2 <= (i / 10) mod 6;
71              d_3 <= (i / 60) mod 10;
72              d_4 <= (i / 600) mod 6;
73              d_5 <= (i / 3600) mod 10;
74              d_6 <= (i / 36000) mod 6;
75              wait for clk_period;
76          end loop;
77
78          -- Wait for simulation to end
79          wait;
80      end process;
81
82  END;
83
```

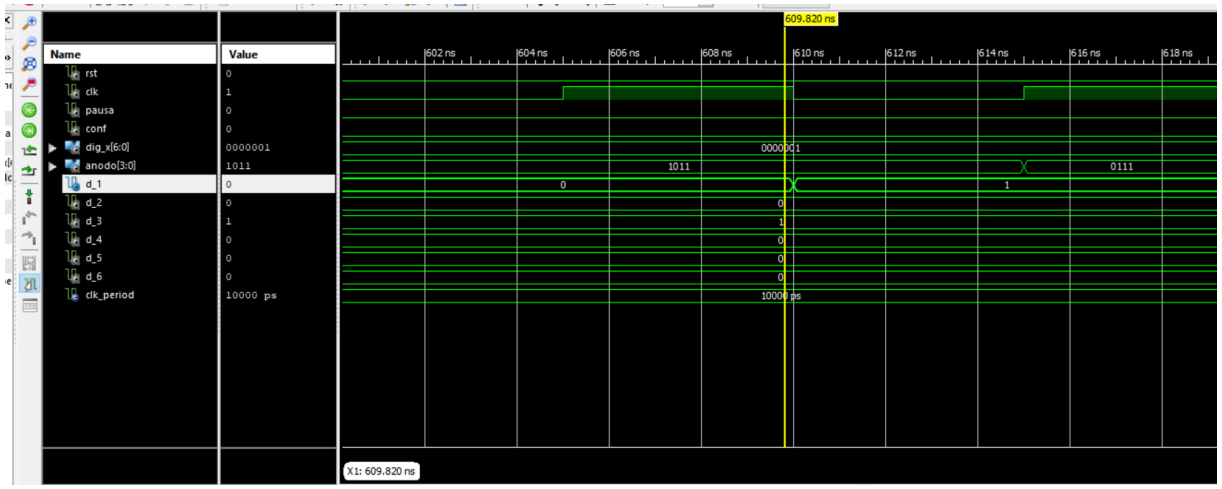
Quando diferentes combinações de entradas são inseridas, os resultados abaixo podem ser obtidos:



Podemos observar que quando d_1 assume o valor “9” o display a esquerda dele “d_2” assume o valor “1”, representando então 10 segundos. Na Figura abaixo é possível observar que quando o d_1 volta a fazer a contagem, o d_2 ainda se mantém no valor “1” até que o d_1 chegue novamente em “9”



o mesmo se repete para os demais displays, como exemplo é possível observar abaixo o d_3 assumindo valor “1”, representando 1 minuto



Abaixo é possível notar que o valor do d_3 se mantém em 1 até que d_2 e d_1 cheguem ao seus respectivos valores de 5 e 9.

Com base no gráfico da simulação e validação executada diretamente na XC3S500E, podemos concluir que o relógio 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:

relogio_module Project Status (05/06/2024 - 18:27:19)			
Project File:	relogio_module.xise	**Parser Errors:**	No Errors
Module Name:	relogio_module	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	32 Warnings (1 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	117	4656	2%
Number of Slice Flip Flops	94	9312	1%
Number of 4 input LUTs	202	9312	2%
Number of bonded IOBs	15	232	6%
Number of GCLKs	1	24	4%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	seg 6. mai 18:27:17 2024	0	32 Warnings (1 new)	3 Infos (0 new)
Translation Report	Out of Date	sex 3. mai 17:10:25 2024	0	0	0
Map Report	Out of Date	sex 3. mai 17:10:28 2024	0	1 Warning (0 new)	2 Infos (0 new)
Place and Route Report	Out of Date	sex 3. mai 17:10:36 2024	0	0	2 Infos (0 new)
Power Report					
Post-PAR Static Timing Report	Out of Date	sex 3. mai 17:10:39 2024	0	0	6 Infos (0 new)
Bitgen Report	Out of Date	sex 3. mai 17:10:52 2024	0	1 Warning (0 new)	0

4 RESULTADOS E CONCLUSÃO:

O projeto de desenvolvimento do cronômetro de horas, minutos e segundos teve como base os conceitos vistos nos laboratórios anteriores e os resultados obtidos foram satisfatórios. Mesmo exibindo quatro dos seis parâmetros definidos (dezenas e unidades de horas, minutos e segundos), o cronômetro se mostrou acurado, visto que ao trocar o nível lógico de *conf* o cronômetro mostrou estar incrementando todos os parâmetros simultaneamente. Além disso, ao pausar e retomar a contagem, o dispositivo obteve sucesso ao retomar de onde parou. Nesse projeto, a linguagem VHDL foi utilizada também para escrever o test bench, permitindo com que fosse possível perceber os erros no meio do processo e corrigi-los antes de implementar o programa 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.