UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIAS
CURSO DE ENGENHARIA ELÉTRICA

LUCAS FILUS RAMOS - GRR20202598
VITÓRIA MARIANA DA ROCHA GASINO - GRR20202564

TE351 DA – LABORATÓRIO 4

CURITIBA
2024

UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Lucas Filus Ramos
Vitória Mariana da Rocha Gasino

TE 351 DA – LABORATÓRIO 3

Relatório técnico apresentado à disciplina Microeletrônica do Curso de Engenharia Elétrica do Setor de Tecnologia da Universidade Federal do Paraná, como parcial para completamento da disciplina.

Orientador: Sibilla Batista da Luz Franca.

CURITIBA

2024

Sumário

1. INTRODUÇÃO

Este relatório tem como objetivo relatar o desenvolvimento de um relógio digital,utilizando VHDL, como parte da disciplina de Microeletrônica I. Nesse caso serão usados os conceitos como processos sequenciais para o monitorando os eventos do clock, bem como a utilização de mais de um componente para atingir o objetivo proposto.

2. PROJETO 1 – Relógio Digital

Para o desenvolvimento do relógio digital, foram desenvolvidos dois componentes(decodificador.vhd e contador.vhd) que foram utilizados na entidade top(lab4_1.vhd).

No ‘contador’, um processo realiza uma contagem a cada 1 Hz, onde quando atingido o valor de clock da placa (definido no código utilizando um *GENERIC)* a contagem incrementa uma unidade à variável ‘*seg_0’ e* é zerado o contador. Quando a variável ‘*seg_0’* atinge o valor 10, é zerado o valor de ‘*seg_0’* e do contador, e incrementa-se uma unidade em *‘seg_1’*, que armazena o valor de dezena dos segundos. E o mesmo processo é repetido para as unidades e dezenas dos minutos e horas, sendo *‘min_0’* incrementada quando *‘seg_1’* atinge o valor 6, *‘min_1’* incrementada quando *‘min_0’* atinge o valor 10, *‘hr_0’* incrementada quando *‘min_1’* atinge o valor 6 e *‘hr_1’* incrementada quando *‘hr_0’* atinge o valor 10. Quando *‘hr_1’* atinge 2 , o código verifica se ‘hr_0’ atingiu 3, caso sim, todos os valores e o contador são zerados. Além disso, esse componente recebe uma *‘flag’* para indicar se o switch de pausa está em valor alto ou baixo. Se estiver em valor alto, significa que o sistema está pausado e a contagem não é realizada.

Ademais, uma condicional no início da sequência da lógica verifica a entrada de reset, e toda vez que for acionada todos os sinais de tempo recebem ‘0’, assim a partir desta condicional um segundo “elsif” inicia com a lógica da contagem de tempo.

No ‘decodificador’ foi realizado a codificação para o display BCD de sete segmentos, assim como realizado nos laboratórios anteriores.

No arquivo top, foi elaborada a indexação dos componentes e mapeamento das portas e dos *generics*, e dentro de um processo, foi realizada a multiplexação para para seleção dos anodos, alternando a cada 1000 ciclos de clock. Uma variável foi criada para que se pudesse alternar entre os anodos, onde a cada ciclo completo de contagem(1000 batidas do clock), o código entra em uma verificação com valor de *'contador_anodo*', verifica e muda para o próximo valor possível de indexação de anodo na base decimal, e ao final atribuí esse valor à saída *'a'*. É importante destacar que foram utilizados os valores na base binária dos valores possíveis para a ativação do anodo, sendo eles : "0111", "1011" , "1101" e "1110".

Além disso, foi solicitado dois modos de exibição: hh:mm e mm:ss , sendo 'hh' o valor de horas contados, 'mm' o valor de minutos e 'ss' o valor de segundos. Para realizar essa operação foi mapeado uma entrada do tipo std_logic, *'modo',* onde caso nível lógico baixo, o vetor de sete bits que o componente de '*d0*' recebe ficará alternando entre *'seg_0','seg_1','min_0'* e *'min_1'* conforme o valor registrado no sinal *'contador_anodo'(*que é alternado a cada 1000 ciclos de clock como explicado no parágrafo anterior).

2.1. Desenvolvimento do código

As figuras a seguir ilustram a entidade top e códigos de cada componente.

FIGURA 1 – Código da entidade top.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
--Por Enquanto sem pausa e sem rst
entity lab4_1 is
    GENERIC( clock_placa: positive := 50); -- 50MHz , 100 para o teste bench
    PORT( clk,modo,pausa,rst: in STD_LOGIC;
          y: out STD_LOGIC_VECTOR(6 downto 0);
          a: out STD_LOGIC_VECTOR(3 downto 0));
end lab4_1;

architecture Behavioral of lab4_1 is
    COMPONENT relogio IS
       GENERIC( clock: positive := 50); --50MHz
       Port ( clk,flag,rst : in std_logic; --flag serve para indicar se a contagem esta pausada ou nao , se 0 nao se 1 sim
              seg_0: out integer range 0 to 9;
              seg_1: out integer range 0 to 5;
              min_0: out integer range 0 to 9;
              min_1: out integer range 0 to 5;
              hr_0: out integer range 0 to 9;
              hr_1: out integer range 0 to 2);
    END COMPONENT;

    COMPONENT decodificador is
       PORT( d : in integer;
          y: out STD_LOGIC_VECTOR(6 downto 0));
    END COMPONENT;

    signal s_int_0: integer range 0 to 9;
    signal s_int_1: integer range 0 to 9;
    signal m_int_0: integer range 0 to 9;
    signal m_int_1: integer range 0 to 9;
    signal h_int_0: integer range 0 to 9;
    signal h_int_1: integer range 0 to 2;
    signal d_0 : integer range 0 to 9;
    signal contador_anodo: std_logic_vector(3 downto 0) := "1110";
begin

    cont_s: relogio GENERIC MAP (clock_placa) PORT MAP (clk,pausa,rst, s_int_0, s_int_1, m_int_0, m_int_1, h_int_0 ,h_int_1);  --contador de segundos


    d0: decodificador PORT MAP (d_0,y);
    d_0 <= s_int_0 WHEN ( modo='0' and contador_anodo="1110") else
          s_int_1 WHEN ( modo='0' and contador_anodo="0111") else
          m_int_0 WHEN ( modo='0' and contador_anodo="1011") else
          m_int_1 WHEN ( modo='0' and contador_anodo="1101") else -- MM :SS

          m_int_0 WHEN ( modo='1' and contador_anodo="1110") else
          m_int_1 WHEN ( modo='1' and contador_anodo="0111") else
          h_int_0 WHEN ( modo='1' and contador_anodo="1011") else
          h_int_1 WHEN ( modo='1' and contador_anodo="1101") else --  HH :SS
          0;
```

```
   process(clk)
   --7    , 11   , 13   , 14
   --0111 , 1011 , 1101 , 1110
   variable contador_x: integer :=0;
   begin

   if(clk'event AND clk='1') then
   contador_x:=contador_x+1;
      if(contador_x=1000) then
         contador_x:=0;
         if(contador_anodo="1110") then
            contador_anodo<="1101";
         elsif(contador_anodo="1101") then
               contador_anodo<="1011";
         elsif(contador_anodo="1011") then
               contador_anodo<="0111";
         elsif(contador_anodo="0111") then
               contador_anodo<="1110";
         end if;
         a<=contador_anodo;
      end if;
   end if;
   end process;


end Behavioral;
```

FONTE: Os autores (2024)

FIGURA 2 – Código do componente “relógio”.

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


entity relogio is
      GENERIC( clock: positive := 100); --50MHz
      Port ( clk,flag,rst : in std_logic; --flag serve para indicar se a contagem esta pausada ou nao , se 0 nao se 1 sim
             seg_0: out integer range 0 to 9;
             seg_1: out integer range 0 to 5;
             min_0: out integer range 0 to 9;
             min_1: out integer range 0 to 5;
             hr_0: out integer range 0 to 9;
             hr_1: out integer range 0 to 2);
end relogio;

architecture Behavioral of relogio is

begin
   process(clk,rst,flag)
      variable contador: INTEGER:=0;
      variable s0, s1, m0,m1,h0,h1: INTEGER:=0;
   begin
      if(rst='1') then
         s0:=0;
         s1:=0;
         m0:=0;
```

```
        ml:=0;
        h0:=0;
        hl:=0;
        contador:=0;
      elsif (clk'event AND clk='1' AND flag='0') then
        contador:=contador+1;
        if (contador = clock) then
            s0:=s0+1;
          if(s0=10) then
            s0:=0;
            sl:= sl+1;
            if(sl=6) then
              sl:= 0;
              m0:=m0+1;
              if(m0=10) then
                m0:=0;
                ml:= ml+1;
                if(ml=6) then
                  ml:= 0;
                  h0:=h0+1;
                  if(h0=10) then
                    h0:=0;
                    hl:=hl+1;
                    if(hl=2 AND h0=3) then
                      hl:=0;
                      h0:=0;
                    end if;
                  end if;
                end if;
              end if;
            end if;
          end if;
          contador:=0;
        end if;
      end if;
   seg_0<=s0;
   seg_1<=sl;
   min_0<=m0;
   min_1<=ml;
   hr_0<=h0;
   hr_1<=hl;
   end process;
end Behavioral;

```

FONTE: Os autores (2024)

FIGURA 3 – Código do Componente “Codificador”.

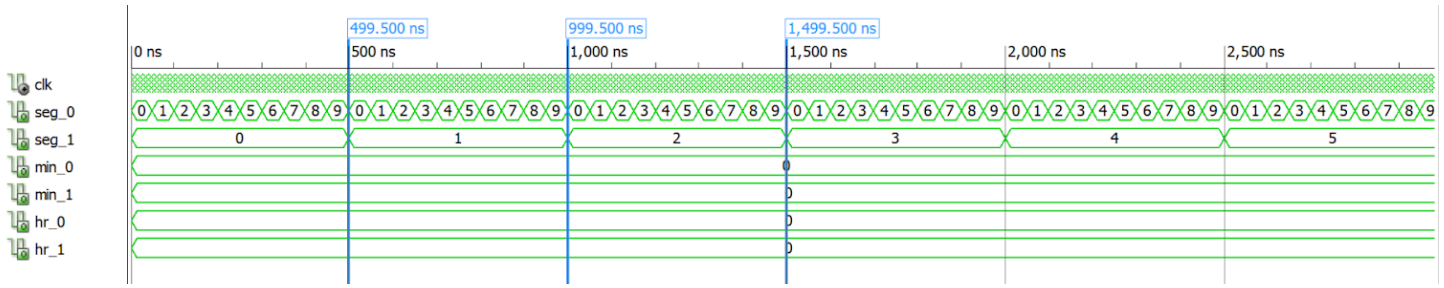
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decodificador is
   PORT( d: in integer;
         y: out STD_LOGIC_VECTOR(6 downto 0));
end decodificador;

architecture Behavioral of decodificador is

begin

   y <= "0000001" when d = 0 else  --0
        "1001111" when d = 1 else  --1
        "0010010" when d = 2 else  --2
        "0000110" when d = 3 else  --3
        "1001100" when d = 4 else  --4
        "0100100" when d = 5 else  --5
        "1100000" when d = 6 else  --6
        "0001111" when d = 7 else  --7
        "0000000" when d = 8 else  --8
        "0001100";                 --9
end Behavioral;
```

FONTE: Os autores (2024)

2.2. Implementação do Test Bench e resultados

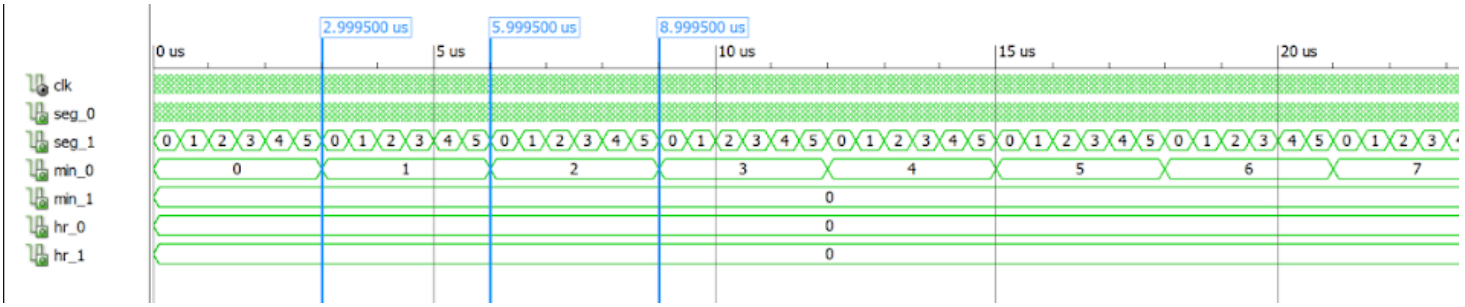
Para avaliar o funcionamento do circuito implementado acima, foi preparado algumas simulações utilizando *clock* em 1 ns e incrementando a cada 50 ciclos, ou seja cada 50 ns representam 1 s . As simulações ilustradas nas Figuras 4 à Figura 6 mostram a incrementação das unidades de contagem de tempo.

FIGURA 4 – Simulação do funcionamento da incrementação da unidade e dezena dos segundos.



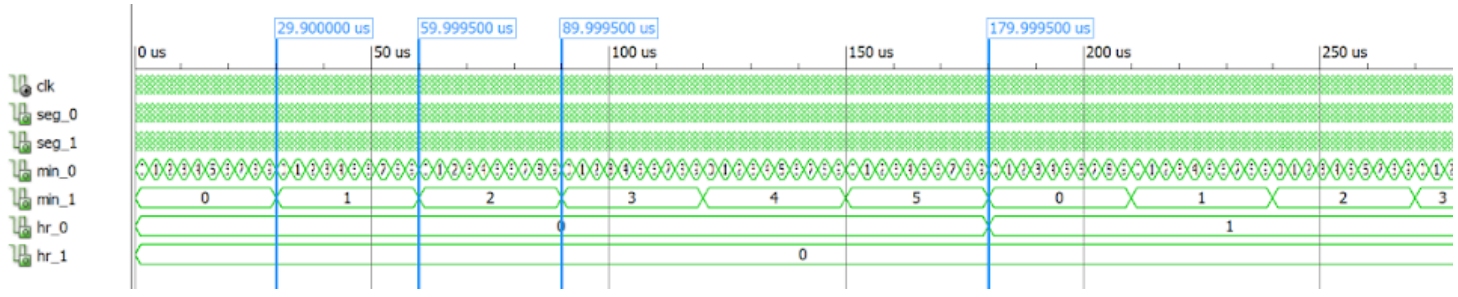
FONTE: Os autores (2024).

FIGURA 5 – Simulação do funcionamento da incrementação da unidade dos minutos.



FONTE: Os autores (2024).

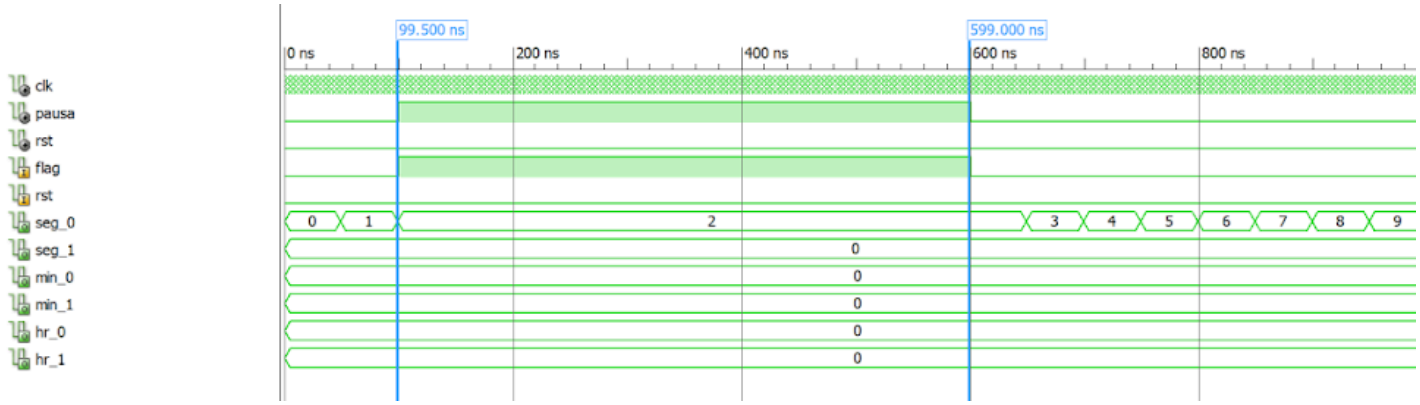
FIGURA 6 – Simulação do funcionamento da incrementação da dezena dos minutos e unidade das horas.



FONTE: Os autores (2024).

A seguir a simulação da Figura 7 demonstra o correto funcionamento da função ‘*pause*’, que quando levada a nível lógico alto interrompe a contagem mas mantêm o tempo que estava sendo marcado e assim que apagado retorna a contagem.

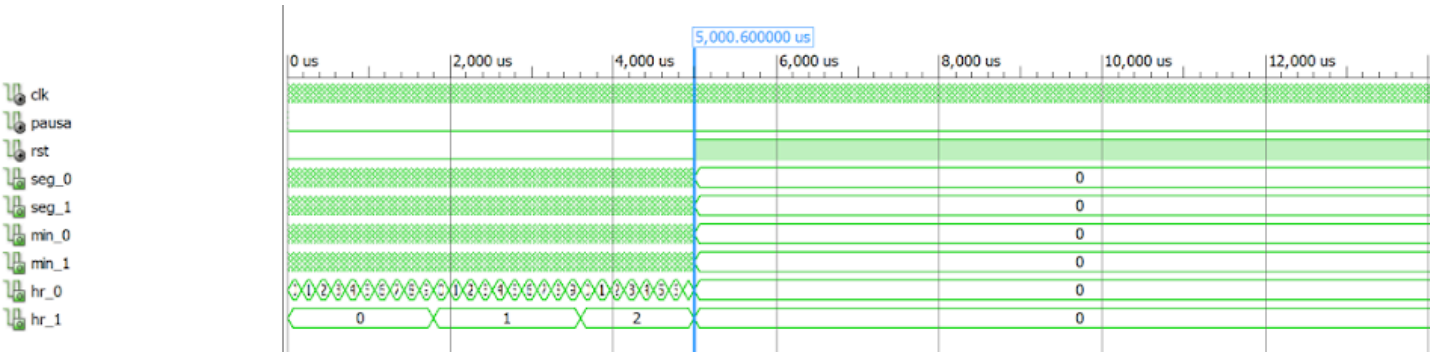
FIGURA 7 – Simulação do funcionamento da função ‘*pause*’.



FONTE: Os autores (2024).

Também foi avaliado o correto funcionamento da função ‘*reset*’ que uma vez acionada retorna a contagem para 0h0min0seg, como mostrado na Figura 8.

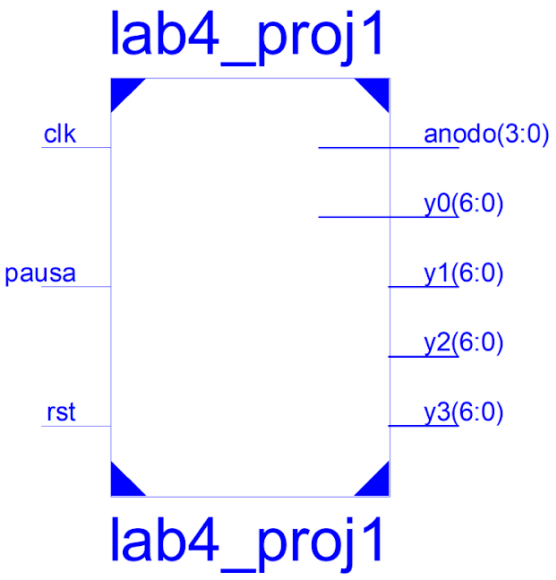
FIGURA 8 – Simulação do funcionamento da função ‘*reset*’.



FONTE: Os autores (2024).

Por fim, é possível avaliar o esquemático do circuito na Figura 9.

FIGURA 8 – Esquemático do circuito.



FONTE: Os autores (2024).

3. CONCLUSÃO

Após a geração dos test bench e implementação no kit NEXYS 2 é possível concluir que, a contagem de eventos de clock se mostra mais uma vez com extrema importância para monitorar o tempo, e nesse caso conseguimos controlar o intervalo temporal a partir desse monitoramento de eventos. O estudo da saída de cada display se mostra relevante já que conseguimos atingir que cada vetor de display representa um valor específico a nível percepção humana. E a utilização de um componente se mostrou com a relevância a nível de código limpo e eficiente.