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

Universidade Tecnológica Federal do Paraná

Engenharia de Computação

IMPLEMENTAÇÃO DE UM TIMER REGRESSIVO COM LED E SSDs EM VHDL

Maria Eduarda Pedroso

Professor orientador: Marcelo de Oliveira

Apucarana

Dezembro/2023

SUMÁRIO

1. RESUMO

Este trabalho apresenta o desenvolvimento de um timer regressivo implementado em uma linguagem de descrição de hardware, VHDL. O sistema conta o tempo regressivamente até atingir zero, acionando um LED e exibindo o tempo restante em displays de 7 segmentos (SSDs). O timer oferece funcionalidades como reset assíncrono, pausa/desabilitação e diferentes opções de tempos configuráveis. O projeto é destinado a hardware específico, como microcontroladores FPGA.

Palavras-chave: Timer regressivo, VHDL, FPGA, LED, Display de 7 segmentos, Double Dabble, Reset assíncrono, Pausa, Hardware específico.

2. INTRODUÇÃO

A contagem regressiva do tempo é uma funcionalidade essencial em diversos contextos, desde dispositivos de uso diário até sistemas mais complexos. Este projeto propõe-se a desenvolver um timer regressivo utilizando a linguagem de descrição de hardware VHDL (VHSIC Hardware Description Language). A concepção desse sistema envolve a aplicação de conceitos fundamentais de design digital, além da implementação de algoritmos de conversão para representação eficiente em displays de 7 segmentos (SSDs).

Além da teoria subjacente à representação numérica, este projeto incorpora aspectos práticos, como a ativação de um LED ao término do tempo. A inclusão de recursos de controle, como botões para reset assíncrono e pausa, visa conferir maior flexibilidade ao usuário no manejo do sistema temporal. A integração eficaz do código VHDL ao hardware específico é um ponto crucial, levando em consideração as particularidades do ambiente de implementação.

Ao explorar esses conceitos teóricos e práticos, busca-se não apenas a criação de um timer regressivo funcional, mas também a construção de uma base sólida para compreensão e aplicação de sistemas temporizadores em ambientes de design digital. Nos capítulos subsequentes, serão detalhados os objetivos específicos, a metodologia empregada e as expectativas em relação aos resultados obtidos, proporcionando uma visão mais profunda e esclarecedora deste trabalho.

2.1. Objetivos

O presente trabalho tem como objetivo:

- Desenvolvimento de um Timer Regressivo: Implementar um timer regressivo em VHDL para contar o tempo regressivamente até zero.
- Ativação de LED: Acionar um LED quando o timer atingir zero, proporcionando uma indicação visual do término do tempo.
- Exibição em Displays de 7 Segmentos (SSDs): Representar o tempo restante nos SSDs, utilizando uma função de conversão.
- Recursos de Controle: Implementar recursos de controle, incluindo um botão para reset assíncrono.
- Opções de Tempo Configuráveis: Incluir suporte para diferentes opções de tempos configuráveis, permitindo ao usuário escolher entre pelo menos duas configurações de tempo.
- Adaptação a Hardware Específico: Garantir a adaptação do código VHDL ao hardware específico utilizado, considerando pinos, portas e requisitos específicos do ambiente.
- Testes e Validação: Realizar testes abrangentes para validar o funcionamento correto do timer, garantindo precisão e confiabilidade nas contagens regressivas.
- Usabilidade e Flexibilidade: Assegurar que o timer seja de fácil uso, permitindo uma configuração intuitiva e flexível para diferentes necessidades de contagem regressiva.

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. Contador

O contador decrescente é implementado no bloco principal do código VHDL. Ele conta de um valor máximo até zero em intervalos de tempo determinados pela frequência do clock. A contagem pode ser iniciada, pausada e reiniciada com base nas condições dos botões de controle, como o botão de reset e o botão de pausa.

```
elsif rising_edge(clk) then
    if counter < StateMachineSteps then
        counter := counter + 1;
    else
        counter := 0;
        if contagem > 0 then
            contagem <= contagem - 1;
            -- Separar o número para os displays
            valor_display1 <= decimal_para_7seg(contagem / 1000); -- Milhar
            valor_display2 <= decimal_para_7seg((contagem mod 1000) /
100); -- Centena
            valor_display3 <= decimal_para_7seg((contagem mod 100) / 10);
-- Dezena
            valor_display4 <= decimal_para_7seg(contagem mod 10); --
Unidade
            LED <= '0';
        else
            contagem <= 0;
            -- Separar o número para os displays
            valor_display1 <= decimal_para_7seg(contagem / 1000); -- Milhar
            valor_display2 <= decimal_para_7seg((contagem mod 1000) /
100); -- Centena
            valor_display3 <= decimal_para_7seg((contagem mod 100) / 10);
-- Dezena
            valor_display4 <= decimal_para_7seg(contagem mod 10); --
Unidade
            LED <= '1';
        end if;
    end if;
```

```
end if;
```

3.1.2. Saída nos displays

Os valores do contador decrescente são exibidos em quatro displays de 7 segmentos. Cada display é responsável por mostrar uma parte específica do número, como milhar, centena, dezena e unidade. A forma que essas partes específicas são extraídas é basicamente contas básicas utilizando o módulo. Abaixo podemos observar essas contas.

```
    function decimal_para_7seg(num : integer) return STD_LOGIC_VECTOR is
    begin
        case num is
            -- Mapeamento dos dígitos para a representação de 7 segmentos
            when 0 => return "11000000";
            when 1 => return "11111001";
            when 2 => return "10100100";
            when 3 => return "10110000";
            when 4 => return "10011001";
            when 5 => return "10010010";
            when 6 => return "10000010";
            when 7 => return "11111000";
            when 8 => return "10000000";
            when 9 => return "10010000";
            when others => return "11111111"; -- caso padrão, exibir "off"
        end case;
    end function;

-- Atualizar valores dos displays
    valor_display1 <= decimal_para_7seg(contagem / 1000); -- Milhar
    valor_display2 <= decimal_para_7seg((contagem mod 1000) / 100); --
Centena
    valor_display3 <= decimal_para_7seg((contagem mod 100) / 10); -- Dezena
    valor_display4 <= decimal_para_7seg(contagem mod 10); -- Unidade

-- Saídas
SSD1 <= valor_display1;
SSD2 <= valor_display2;
SSD3 <= valor_display3;
SSD4 <= valor_display4;
```

3.1.3. Botão reset

O botão de reset reinicia o contador para o valor máximo predefinido. Isso permite reiniciar a contagem a partir do início, independentemente da fase em que a contagem esteja. Temos duas opções iniciais de valores.

```
-- Botão Reset
if rst = '0' then
    if sw = "00" then
        contagem <= CONTAGEM_MAX_1;
    else
        contagem <= CONTAGEM_MAX_2;
    end if;
```

3.1.4. Funcionalidade pause

O botão de pausa permite interromper temporariamente a contagem. Quando pressionado, a contagem é pausada, e o contador mantém seu valor atual. A contagem só é retomada quando o botão de pausa é liberado.

```
-- Botão Pause
if pause = '0' then
    -- Lógica para pausar a contagem
else
    -- Lógica para retomar a contagem
end if;
```

3.1.5. Led

Um LED é utilizado para indicar visualmente o término da contagem. Quando o contador atinge zero, o LED é aceso, sinalizando que a contagem chegou ao fim.

```
-- LED
LED <= '1' when contagem = 0 else '0';
```

3.1.6. PinPlanner

Abaixo podemos observar todos os pinos utilizados nesse projeto.

Figura 1 - PinPlanner.

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
btn_start	Input	PIN_B8	7	B7_N0	PIN_B8	2.5 V		12mA (default)			
clk	Input	PIN_P11	3	B3_N0	PIN_P11	2.5 V		12mA (default)			
LED	Output	PIN_B11	7	B7_N0	PIN_B11	2.5 V		12mA (default)	2 (default)		
pause	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)			
rst	Input	PIN_A7	7	B7_N0	PIN_A7	2.5 V		12mA (default)			
SSD1[7]	Output	PIN_D22	6	B6_N0	PIN_D22	2.5 V		12mA (default)	2 (default)		
SSD1[6]	Output	PIN_E17	6	B6_N0	PIN_E17	2.5 V		12mA (default)	2 (default)		
SSD1[5]	Output	PIN_D19	6	B6_N0	PIN_D19	2.5 V		12mA (default)	2 (default)		
SSD1[4]	Output	PIN_C20	6	B6_N0	PIN_C20	2.5 V		12mA (default)	2 (default)		
SSD1[3]	Output	PIN_C19	7	B7_N0	PIN_C19	2.5 V		12mA (default)	2 (default)		
SSD1[2]	Output	PIN_E21	6	B6_N0	PIN_E21	2.5 V		12mA (default)	2 (default)		
SSD1[1]	Output	PIN_E22	6	B6_N0	PIN_E22	2.5 V		12mA (default)	2 (default)		
SSD1[0]	Output	PIN_F21	6	B6_N0	PIN_F21	2.5 V		12mA (default)	2 (default)		
SSD2[7]	Output	PIN_A19	7	B7_N0	PIN_A19	2.5 V		12mA (default)	2 (default)		
SSD2[6]	Output	PIN_B22	6	B6_N0	PIN_B22	2.5 V		12mA (default)	2 (default)		
SSD2[5]	Output	PIN_C22	6	B6_N0	PIN_C22	2.5 V		12mA (default)	2 (default)		
SSD2[4]	Output	PIN_B21	6	B6_N0	PIN_B21	2.5 V		12mA (default)	2 (default)		
SSD2[3]	Output	PIN_A21	6	B6_N0	PIN_A21	2.5 V		12mA (default)	2 (default)		
SSD2[2]	Output	PIN_B19	7	B7_N0	PIN_B19	2.5 V		12mA (default)	2 (default)		
SSD2[1]	Output	PIN_A20	7	B7_N0	PIN_A20	2.5 V		12mA (default)	2 (default)		
SSD2[0]	Output	PIN_B20	6	B6_N0	PIN_B20	2.5 V		12mA (default)	2 (default)		
SSD3[7]	Output	PIN_A16	7	B7_N0	PIN_A16	2.5 V		12mA (default)	2 (default)		
SSD3[6]	Output	PIN_B17	7	B7_N0	PIN_B17	2.5 V		12mA (default)	2 (default)		
SSD3[5]	Output	PIN_A18	7	B7_N0	PIN_A18	2.5 V		12mA (default)	2 (default)		
SSD3[4]	Output	PIN_A17	7	B7_N0	PIN_A17	2.5 V		12mA (default)	2 (default)		
SSD3[3]	Output	PIN_B16	7	B7_N0	PIN_B16	2.5 V		12mA (default)	2 (default)		
SSD3[2]	Output	PIN_E18	6	B6_N0	PIN_E18	2.5 V		12mA (default)	2 (default)		
SSD3[1]	Output	PIN_D18	6	B6_N0	PIN_D18	2.5 V		12mA (default)	2 (default)		
SSD3[0]	Output	PIN_C18	7	B7_N0	PIN_C18	2.5 V		12mA (default)	2 (default)		
SSD4[7]	Output	PIN_D15	7	B7_N0	PIN_D15	2.5 V		12mA (default)	2 (default)		
SSD4[6]	Output	PIN_C17	7	B7_N0	PIN_C17	2.5 V		12mA (default)	2 (default)		
SSD4[5]	Output	PIN_D17	7	B7_N0	PIN_D17	2.5 V		12mA (default)	2 (default)		
SSD4[4]	Output	PIN_E16	7	B7_N0	PIN_E16	2.5 V		12mA (default)	2 (default)		
SSD4[3]	Output	PIN_C16	7	B7_N0	PIN_C16	2.5 V		12mA (default)	2 (default)		

Fonte: Autoria Própria

3.2. Código VHDL

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

```
-- Declaração de bibliotecas e pacotes IEEE
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Declaração da entidade
entity aula230830 is
    generic (
        ClockFrequencyHz : integer := 50_000_000  -- Frequência do clock padrão é 50 MHz
    );
    Port (
        clk, rst, pause, btn_start : in STD_LOGIC;
```

```
        sw : in STD_LOGIC_VECTOR(1 downto 0);
        LED : out STD_LOGIC;
        SSD1 : out STD_LOGIC_VECTOR(7 downto 0);
                    SSD2 : out STD_LOGIC_VECTOR(7 downto 0);
                    SSD3 : out STD_LOGIC_VECTOR(7 downto 0);
                    SSD4 : out STD_LOGIC_VECTOR(7 downto 0)
    );
end aula230830;

-- Implementação da arquitetura
architecture Behavioral of aula230830 is
    -- Declaração de sinais e constantes
    signal contagem : integer := 0;
    signal valor_display1 : STD_LOGIC_VECTOR(7 downto 0) := "00000000";
            signal valor_display2 : STD_LOGIC_VECTOR(7 downto 0) := "00000000";
            signal valor_display3 : STD_LOGIC_VECTOR(7 downto 0) := "00000000";
            signal valor_display4 : STD_LOGIC_VECTOR(7 downto 0) := "00000000";
    constant CONTAGEM_MAX_1 : integer := 9040; -- Valor máximo para contar durante 10
segundos
    constant CONTAGEM_MAX_2 : integer := 5030; -- Valor máximo para contar durante 20
segundos

             -- Função para converter um número decimal para a representação de 7
segmentos
    function decimal_para_7seg(num : integer) return STD_LOGIC_VECTOR is
    begin
        case num is
            -- Mapeamento dos dígitos para a representação de 7 segmentos
            when 0 => return "11000000";
            when 1 => return "11111001";
            when 2 => return "10100100";
            when 3 => return "10110000";
            when 4 => return "10011001";
            when 5 => return "10010010";
            when 6 => return "10000010";
            when 7 => return "11111000";
            when 8 => return "10000000";
            when 9 => return "10010000";
            when others => return "11111111"; -- caso padrão, exibir "off"
        end case;
    end function;

begin
    -- Processo principal
    process(clk, rst, btn_start)
        variable counter : integer := 0;  -- Variável para contar os passos da máquina de
estados
```

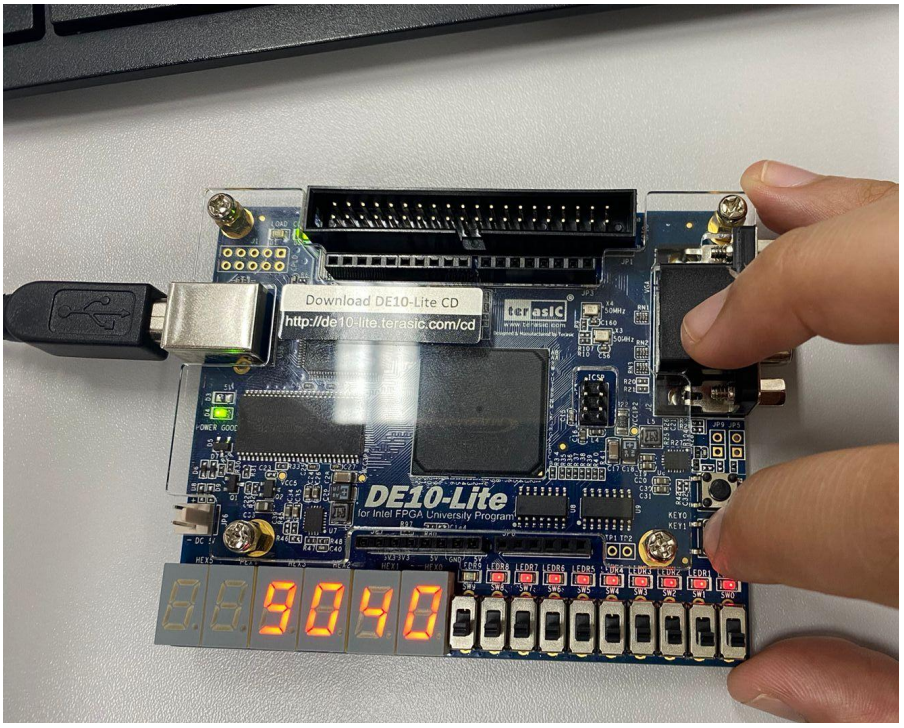
```
      variable StateMachineSteps : integer := ClockFrequencyHz;  -- Passos da máquina
de estados
   begin
      if pause = '0' then
         if rst = '0' then
            -- Seleciona o valor máximo com base na chave SW
            if sw = "00" then
               contagem <= CONTAGEM_MAX_1;
            else
               contagem <= CONTAGEM_MAX_2;
            end if;
            -- Separar o número para os displays
                                          valor_display1 <= decimal_para_7seg(contagem /
1000); -- Milhar
                                          valor_display2 <= decimal_para_7seg((contagem
mod 1000) / 100); -- Centena
                                          valor_display3 <= decimal_para_7seg((contagem
mod 100) / 10); -- Dezena
                                          valor_display4 <= decimal_para_7seg(contagem mod
10); -- Unidade

            LED <= '0';
         elsif rising_edge(clk) then
            if counter < StateMachineSteps then
               counter := counter + 1;
            else
               counter := 0;
               if contagem > 0 then
                  contagem <= contagem - 1;

                                                      -- Separar o número para os
displays
                                                            valor_display1 <=
decimal_para_7seg(contagem / 1000); -- Milhar
                                                            valor_display2 <=
decimal_para_7seg((contagem mod 1000) / 100); -- Centena
                                                            valor_display3 <=
decimal_para_7seg((contagem mod 100) / 10); -- Dezena
                                                            valor_display4 <=
decimal_para_7seg(contagem mod 10); -- Unidade
                  LED <= '0';
               else
                  contagem <= 0;
                  -- Separar o número para os displays
                                                            valor_display1 <=
decimal_para_7seg(contagem / 1000); -- Milhar
                                                            valor_display2 <=
```

```
decimal_para_7seg((contagem mod 1000) / 100); -- Centena
                                                    valor_display3 <=
decimal_para_7seg((contagem mod 100) / 10); -- Dezena
                                                    valor_display4 <=
decimal_para_7seg(contagem mod 10); -- Unidade
                    LED <= '1';
                end if;
            end if;
        end if;
    end if;
  end process;

  -- Saídas
  SSD1 <= valor_display1;
        SSD2 <= valor_display2;
        SSD3 <= valor_display3;
        SSD4 <= valor_display4;

end Behavioral;
```

4. RESULTADOS E DISCUSSÃO

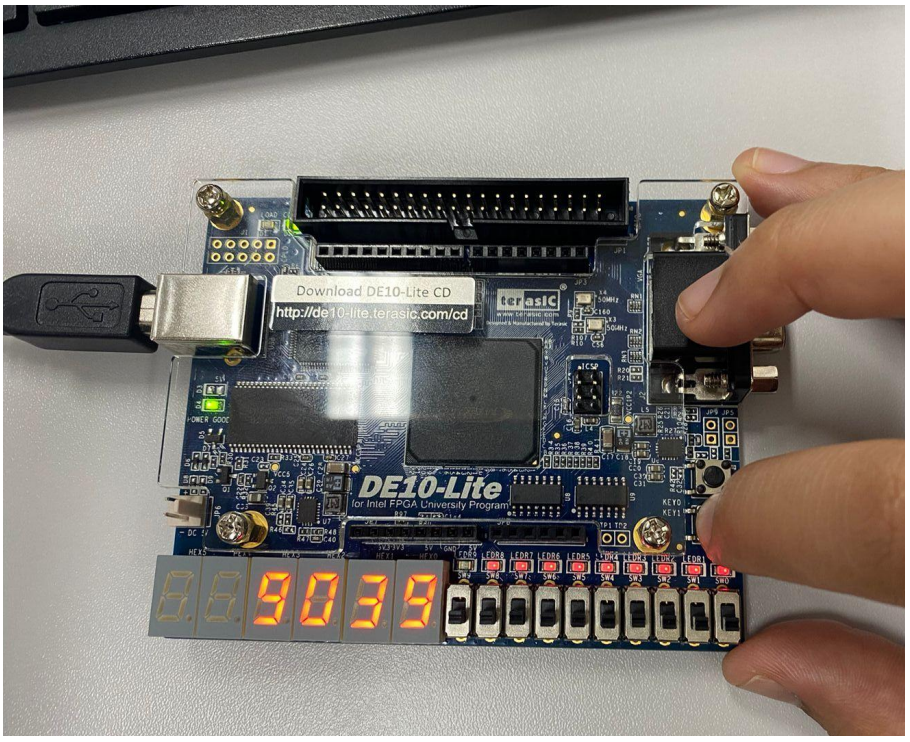
Com a implementação funcionando utilizamos da plaquinha FPGA para ver os resultados das saídas, como podemos observar o contador está decremento, o botão de reset também funcionando como podemos observar na figura 4, e o pause está sendo efetivo travando o contador e só retornando após essa funcionalidade ser desabilitada, na figura 6, 7, 8 e 9 conseguimos comprovar essa afirmação.

Figura 2 - Início máquina.



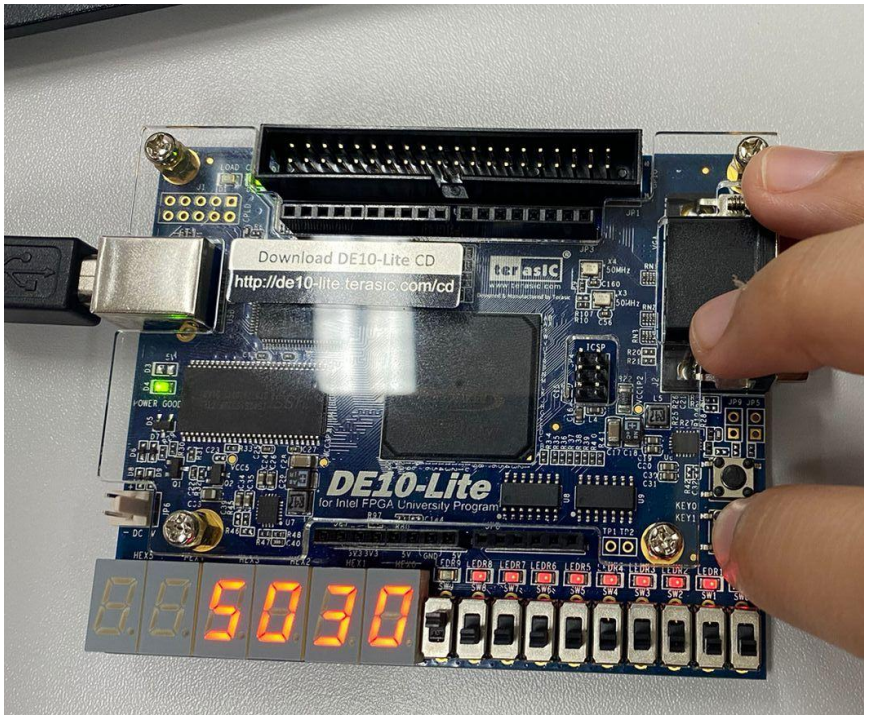
Fonte: Autoria Própria

Figura 3 - Máquina decrementando.



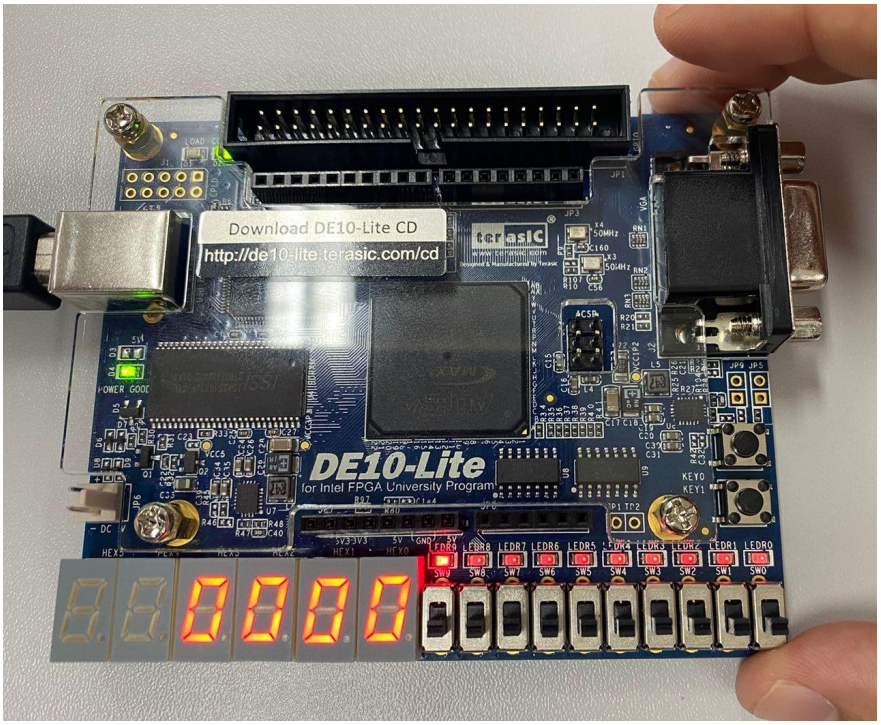
Fonte: Autoria Própria

Figura 4 - Resetando com o segundo número.



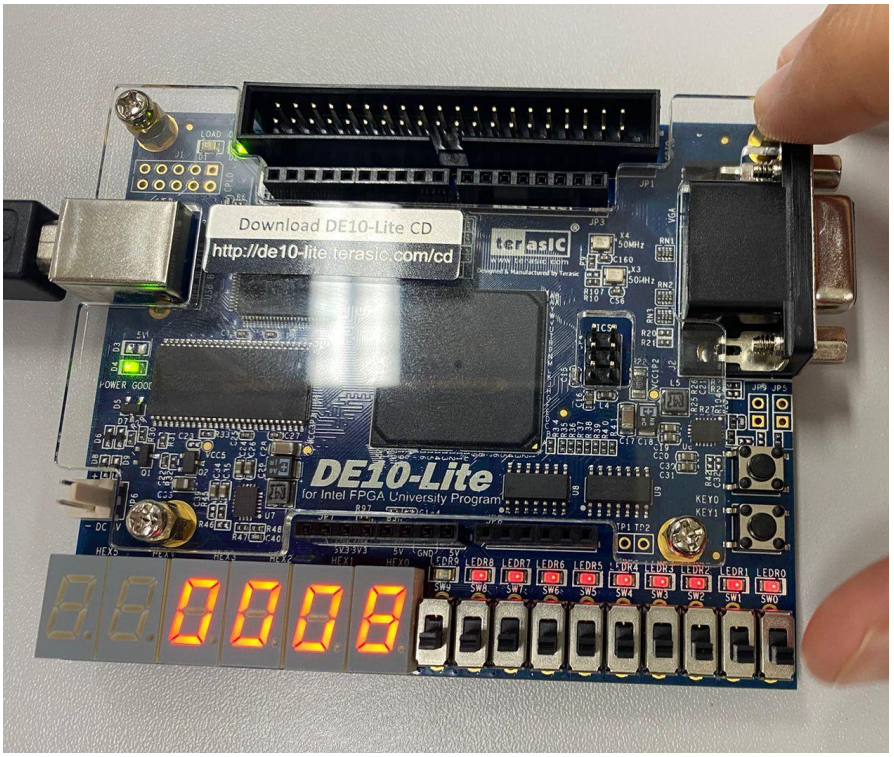
Fonte: Autoria Própria

Figura 5 - Estado final.



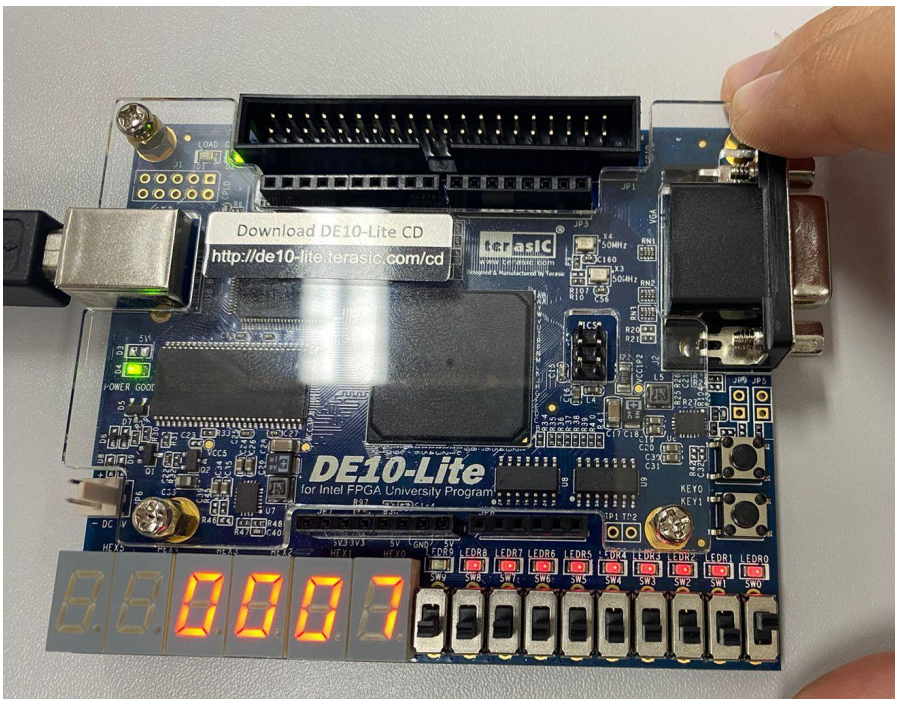
Fonte: Autoria Própria

Figura 6 - Decremento em 8.



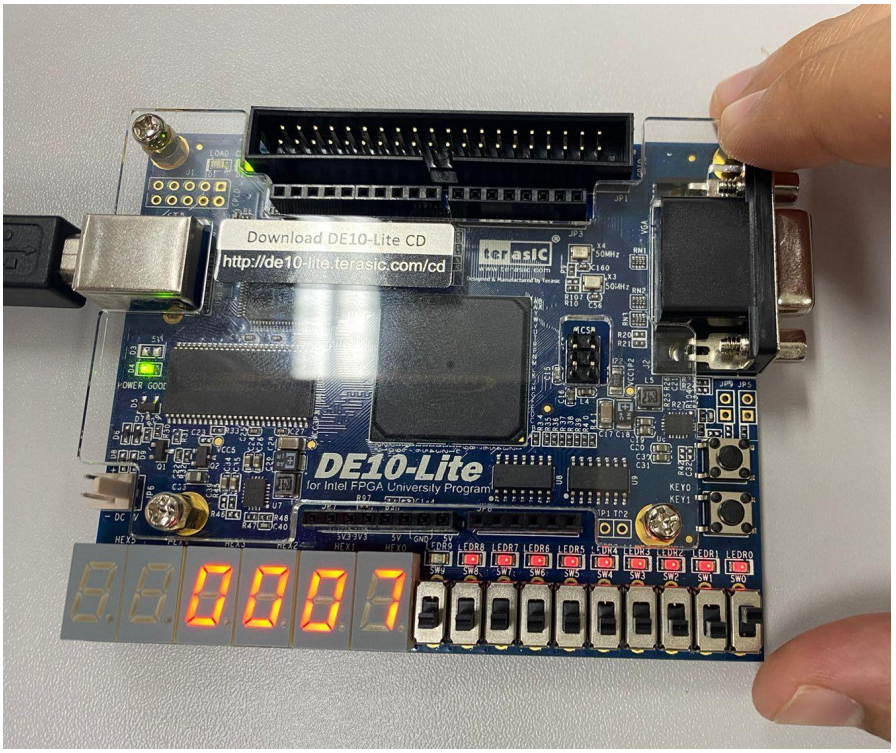
Fonte: Autoria Própria

Figura 7 - Botão pause Ativo.



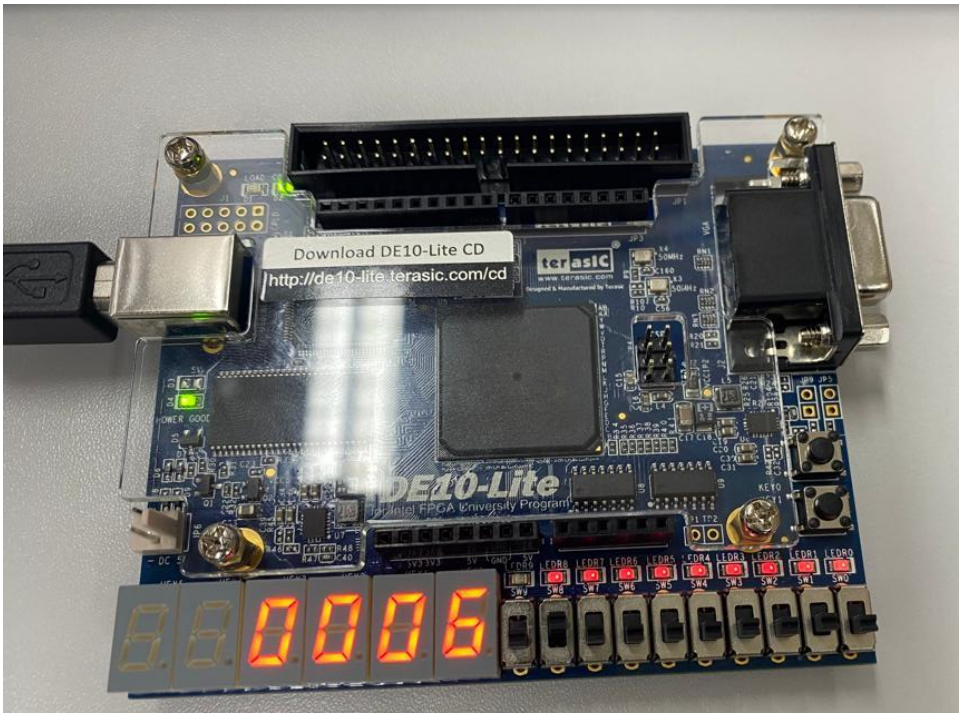
Fonte: Autoria Própria

Figura 8 - Botão pause ativo ainda.



Fonte: Autoria Própria

Figura 9 - Botão pause desativo, maquina decrementando.



Fonte: Autoria Própria

5. CONSIDERAÇÕES FINAIS

O desenvolvimento e implementação do timer regressivo utilizando VHDL representam um marco neste projeto, utilizando conceitos teóricos fundamentais de design digital com a aplicação prática de algoritmos de conversão. A contagem regressiva, apresentada de maneira clara nos displays de 7 segmentos, reforça a importância da representação numérica eficiente para uma interação intuitiva com o usuário.

A incorporação de recursos de controle, como o botão para reset assíncrono, adiciona uma camada de versatilidade ao sistema, proporcionando ao usuário maior flexibilidade no gerenciamento do tempo. A ativação do LED ao término da contagem reforça a comunicação visual do estado do timer, contribuindo para uma experiência de usuário mais completa.

É crucial destacar a adaptação do código VHDL ao hardware específico, considerando requisitos como pinagem e portas. Essa integração bem-sucedida é um testemunho da aplicação prática dos princípios teóricos desenvolvidos ao longo do projeto.

Por fim, este trabalho não apenas cumpre o propósito de criar um timer regressivo funcional, mas também estabelece uma base sólida para a compreensão de sistemas temporizadores em ambientes de design digital. O conhecimento adquirido ao longo deste projeto contribui para a formação técnica e promove uma abordagem holística na concepção e implementação de soluções temporais.