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 MÁQUINA DE ESTADOS EM 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 código VHDL inclui uma função de conversão para conseguir inserir os números nos displays. O timer oferece funcionalidades como reset assíncrono, com 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, Reset assíncrono, 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, 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 e um botão para pausar/desabilitar temporariamente o timer.
- 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.
- Maquina de estados: Entender o funcionamento e estados da aplicação para obter um resultado significativo.

3. METODOLOGIA

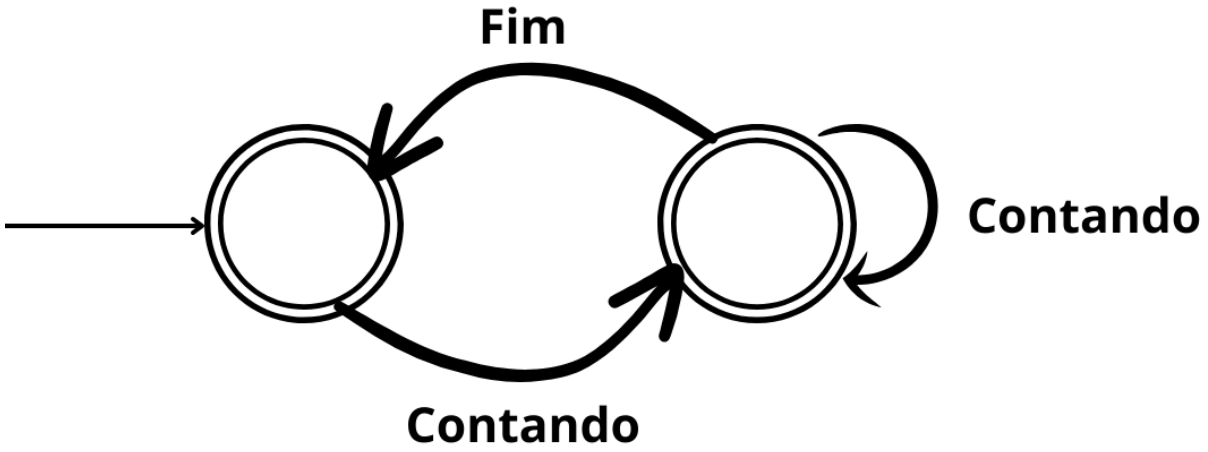
O presente trabalho foi desenvolvido utilizando os conceitos e técnicas estudados durante as aulas da disciplina. Bem como foi empregado o uso do software Quartus, de modo a facilitar e simular nossa lógica.

3.1. Funcionalidades do projeto

3.1.1. Máquina de estados

A implementação do projeto incorpora uma Máquina de Estados para controlar as diferentes fases de operação do sistema. A máquina possui dois estados principais: CONTANDO e FIM. Abaixo conseguimos visualizar e entender seu diagrama.

Figura 1 - Máquina de estados.



Fonte: Autoria Própria

Durante o estado CONTANDO, o sistema decrementa a contagem de tempo, atualiza os valores dos displays de sete segmentos e controla o LED para indicar que o contador está ativo. Quando a contagem atinge zero, o

sistema transita para o estado FIM, onde o LED é aceso, indicando que o tempo chegou ao fim. A estrutura da máquina de estados é vital para a coordenação eficaz das operações do sistema.

```
type estado_tipo is (CONTANDO, FIM);
signal estado : estado_tipo := FIM;
```

3.1.2. Contador

O componente de Contador é responsável por gerenciar o tempo no sistema. No estado CONTANDO, a variável 'contagem' é decrementada a cada ciclo de clock. O valor atualizado é então convertido em representações de sete segmentos para cada dígito e exibido nos SSDs correspondentes. Este módulo é crucial para o controle preciso da contagem regressiva.

```
if counter < StateMachineSteps then
   counter := counter + 1;
else
   counter := 0;
   case estado is
      when CONTANDO =>
         if contagem > 0 then
            contagem <= contagem - 1;
            -- Atualiza os valores dos displays de sete segmentos
            valor_display1 <= decimal_para_7seg(contagem / 1000);
            valor_display2 <= decimal_para_7seg((contagem mod 1000) /
100);
            valor_display3 <= decimal_para_7seg((contagem mod 100) /
10);
            valor_display4 <= decimal_para_7seg(contagem mod 10);
            LED <= '0';
            estado <= CONTANDO;
         else
            -- Quando a contagem atinge zero, transita para o estado FIM
            contagem <= 0;
            valor_display1 <= decimal_para_7seg(contagem / 1000);
            valor_display2 <= decimal_para_7seg((contagem mod 1000) /
```

```
100);
            valor_display3 <= decimal_para_7seg((contagem mod 100) /
10);
            valor_display4 <= decimal_para_7seg(contagem mod 10);
            estado <= FIM;
            LED <= '1';
         end if;
      when others =>
         null;
   end case;
end if;
```

3.1.3. Saída nos displays

As saídas nos displays são atualizadas durante o estado CONTANDO. Os valores dos dígitos individuais da contagem são convertidos para suas representações equivalentes de sete segmentos. Cada dígito é então atribuído aos SSDs correspondentes, proporcionando uma visualização clara da contagem regressiva.

```
valor_display1 <= decimal_para_7seg(contagem / 1000);
valor_display2 <= decimal_para_7seg((contagem mod 1000) / 100);
valor_display3 <= decimal_para_7seg((contagem mod 100) / 10);
valor_display4 <= decimal_para_7seg(contagem mod 10);
```

3.1.4. Botão reset

O botão de reset (rst) é utilizado para reiniciar o sistema. Quando o botão é pressionado, o sistema é reiniciado, definindo a contagem para o valor máximo correspondente à chave SW. Os displays de sete segmentos e o LED são atualizados de acordo, garantindo uma reinicialização eficiente e controlada.

```
if rst = '0' then
   if sw = "00" then
      contagem <= CONTAGEM_MAX_1;
   else
```

```
        contagem <= CONTAGEM_MAX_2;
    end if;
    valor_display1 <= decimal_para_7seg(contagem / 1000);
    valor_display2 <= decimal_para_7seg((contagem mod 1000) / 100);
    valor_display3 <= decimal_para_7seg((contagem mod 100) / 10);
    valor_display4 <= decimal_para_7seg(contagem mod 10);
    estado <= CONTANDO;
    LED <= '0';
end if;
```

3.1.5. Funcionalidade pause

A funcionalidade de pausa (pause) é incorporada para permitir a suspensão temporária do contador. Quando o sinal de pausa é ativado, o contador de tempo é congelado, mantendo seu estado atual até que a pausa seja desativada. Isso proporciona flexibilidade e controle ao usuário sobre a execução do contador.

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

3.1.6. Led

O LED é utilizado como indicador visual do estado do sistema. Durante o estado CONTANDO, o LED permanece apagado ('0'). Ao atingir zero, o LED é aceso ('1'), indicando que a contagem chegou ao fim. Essa representação visual é fundamental para fornecer feedback imediato ao usuário sobre o estado atual do sistema.

```
LED <= '0'; -- LED apagado durante a contagem
LED <= '1'; -- LED aceso quando a contagem atinge zero
```

3.1.7. PinPlanner

Abaixo podemos observar todos os pinos utilizados nesse projeto.

Figura 2 - 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 FREQUENCIA_CLK : integer := 50_000_000; -- Frequência do clock em Hz
   constant PRESCALADOR : integer := 1_000; -- Fator de divisão do clock para contar em
milissegundos

   type estado_tipo is (CONTANDO, FIM);
   signal estado : estado_tipo := FIM;

   constant CONTAGEM_MAX_1 : integer := 9; -- Valor máximo para contar durante 10
segundos
   constant CONTAGEM_MAX_2 : integer := 5; -- Valor máximo para contar durante 20
segundos

   signal contagem_max : integer;

   -- 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
        estado <= CONTANDO;
                                           LED <= '0';

      elsif rising_edge(clk) then
        if counter < StateMachineSteps then
          counter := counter + 1;
        else
          counter := 0;
          case estado is
            when CONTANDO =>
              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
```

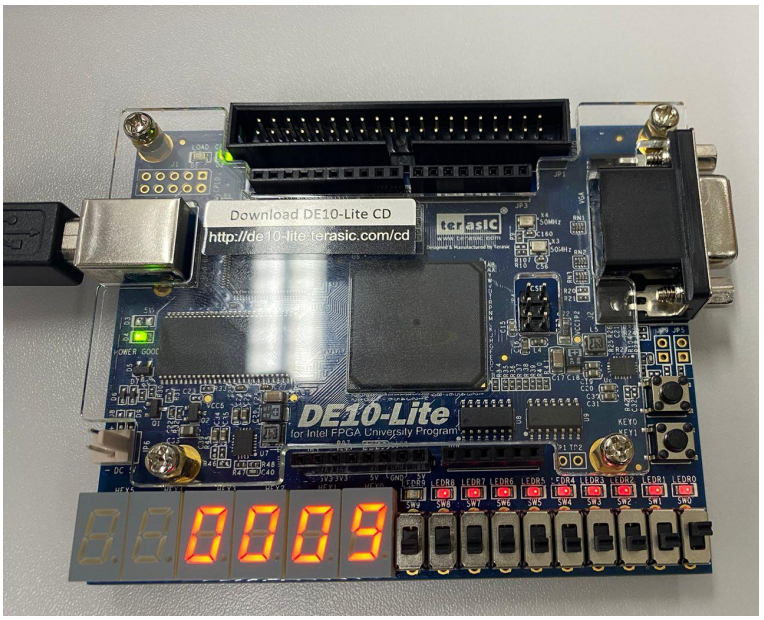
```
<= decimal_para_7seg(contagem mod 10); -- Unidade
                    LED <= '0';
                                                                    estado <=
CONTANDO;
                  else
                    contagem <= 0;
                    -- Separar o número para os displays
                                                                    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
                    estado <= FIM;
                    LED <= '1';
                  end if;
                when others =>
                  null;
              end case;
            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 a máquina funcionou adequadamente, conseguindo realizar as mesmas funcionalidades que nosso código sem máquina, o contador está decremento, o botão de reset também funcionando

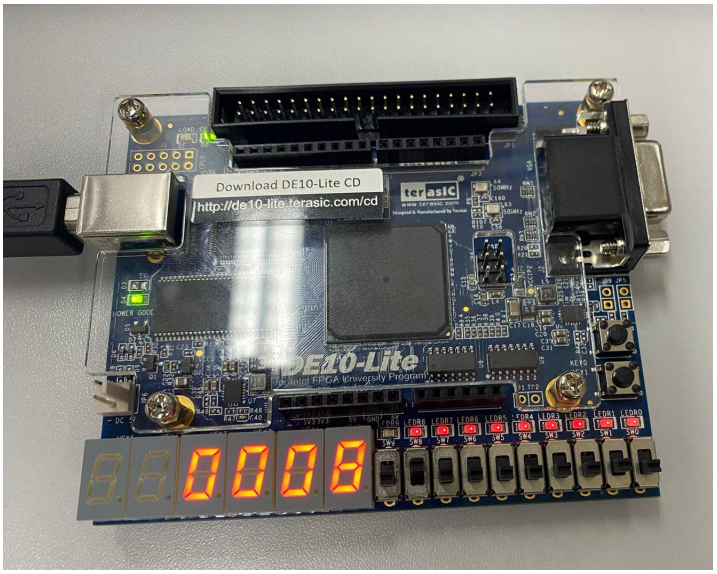
como podemos observar abaixo, e o pause está sendo efetivo travando o contador e só retornando após essa funcionalidade ser desabilitada.

Figura 3 - Início da máquina.



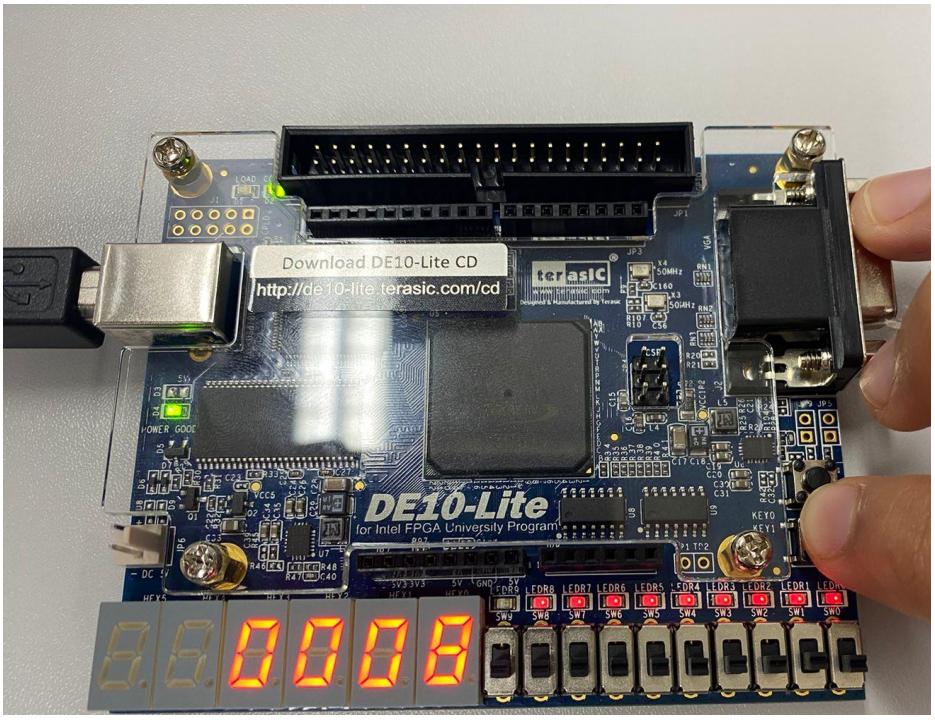
Fonte: Autoria Própria

Figura 4 - Máquina decrescendo.



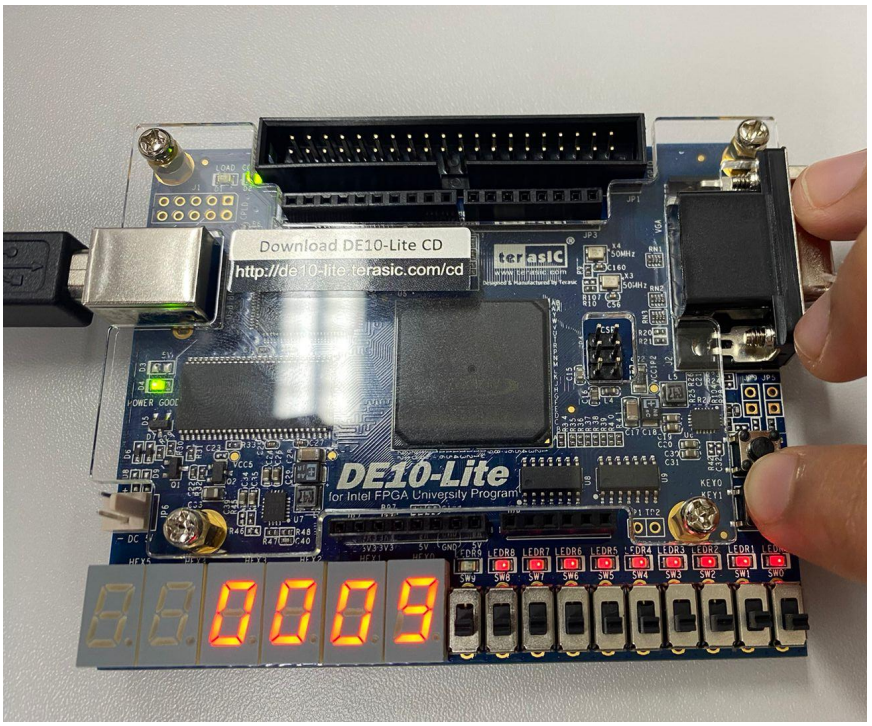
Fonte: Autoria Própria

Figura 5 - Máquina decrescendo.



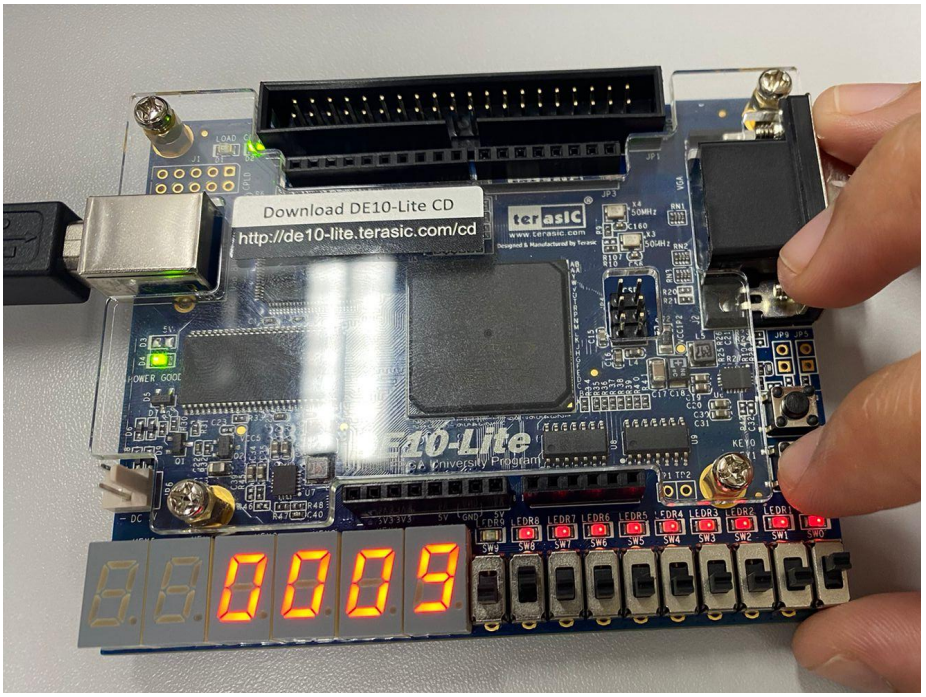
Fonte: Autoria Própria

Figura 6 - Botão reset.



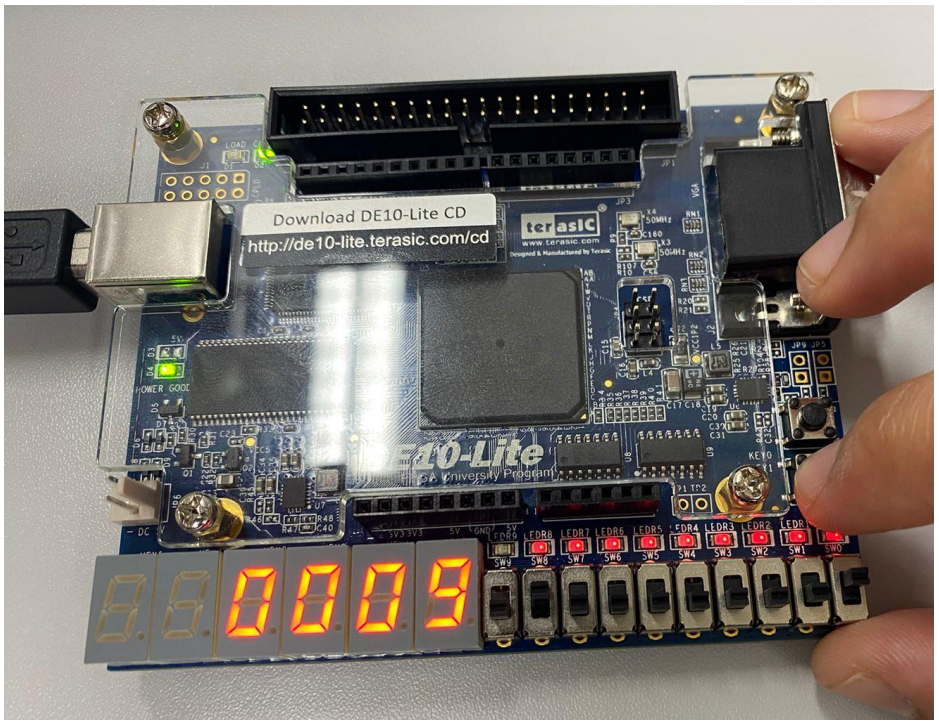
Fonte: Autoria Própria

Figura 7 - Voltando à contagem, botão pressionado.



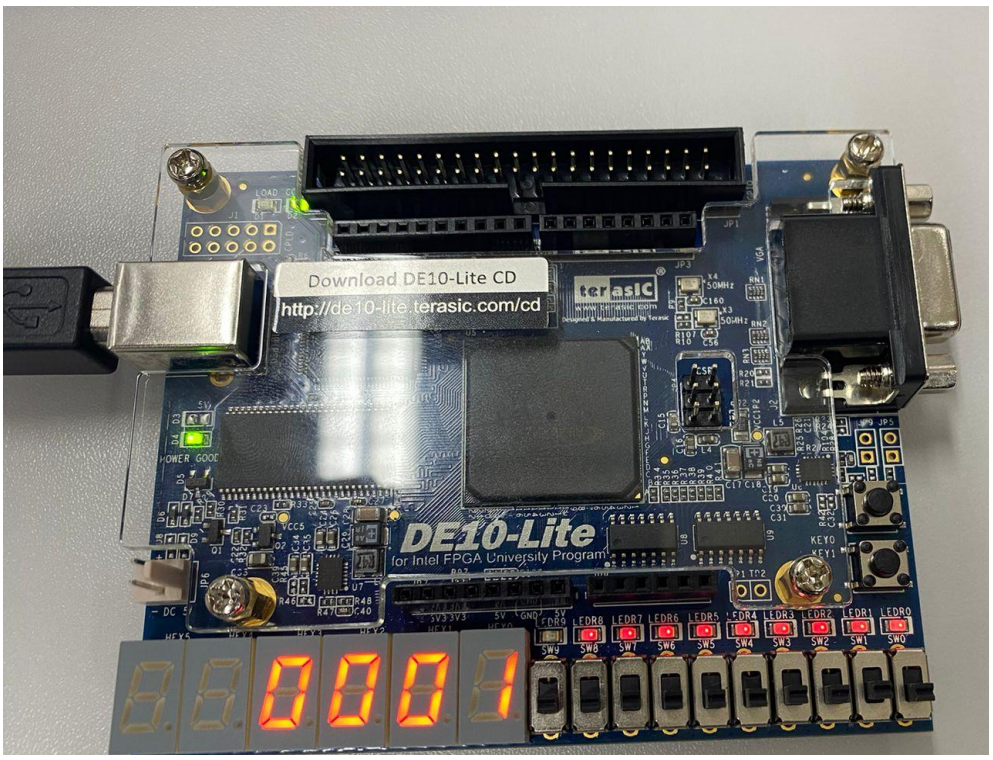
Fonte: Autoria Própria

Figura 8 - Botão despressionado.



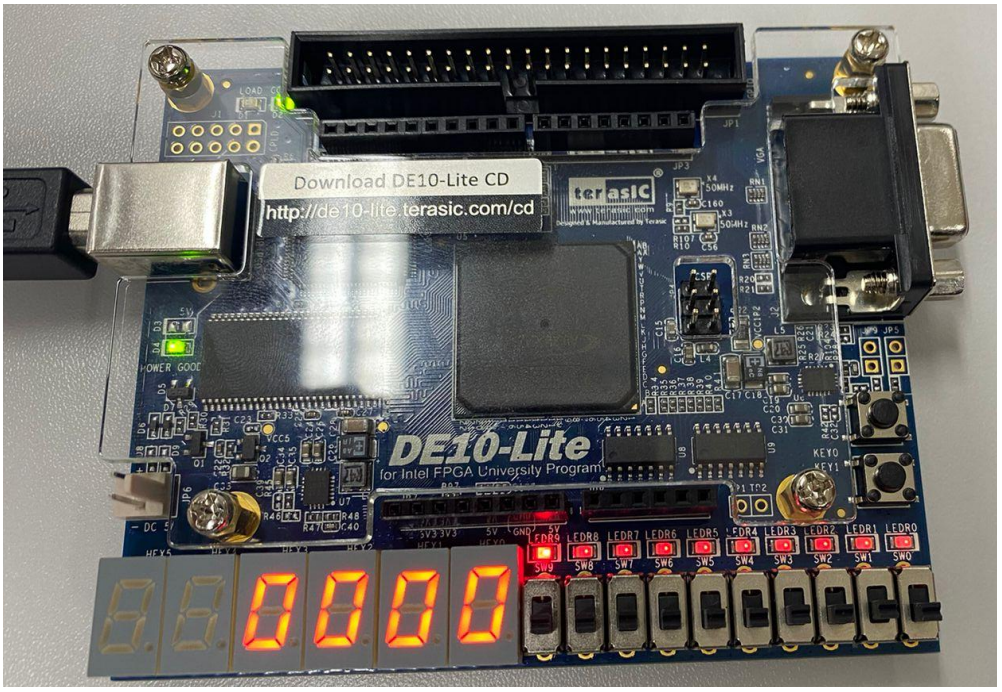
Fonte: Autoria Própria

Figura 9 - Contagem em 1.



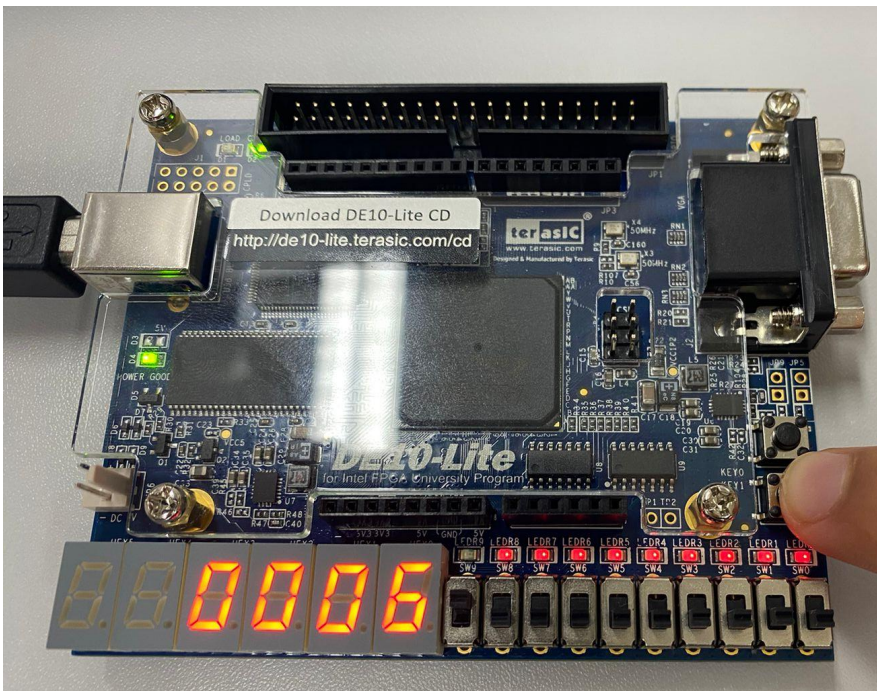
Fonte: Autoria Própria

Figura 10 - Finalizando contagem.



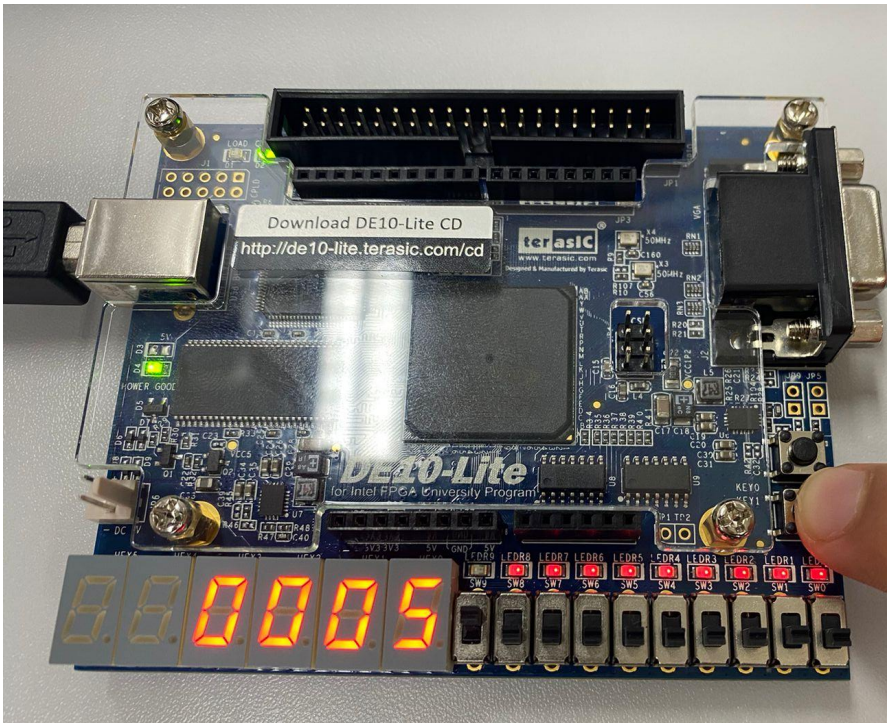
Fonte: Autoria Própria

Figura 11 - Contagem em 6.



Fonte: Autoria Própria

Figura 12 - Botão reset.



Fonte: Autoria Própria

5. CONSIDERAÇÕES FINAIS

As considerações finais deste projeto revelam a eficiência da implementação do contador regressivo utilizando a linguagem VHDL e a abordagem de máquinas de estados. A estrutura modular do projeto facilita a gestão das funcionalidades do sistema.

A máquina de estados, composta pelos estados CONTANDO e FIM, demonstra um controle preciso sobre as fases do contador. Durante o estado CONTANDO, a contagem de tempo é decrementada, os valores nos displays de sete segmentos são atualizados e o LED permanece apagado. Ao atingir zero, a transição para o estado FIM ocorre, acendendo o LED para indicar o término da contagem.

O componente de contador desempenha um papel central, gerenciando o tempo e controlando as atualizações visuais nos displays. A representação intuitiva nos displays facilita a compreensão do usuário sobre o tempo restante.

Funcionalidades adicionais, como o botão de reset, permitem uma reinicialização controlada do sistema, ajustando a contagem inicial de acordo com a posição da chave SW. A capacidade de pausar o contador proporciona flexibilidade ao usuário.

O LED, atuando como indicador visual, oferece feedback imediato sobre o estado do sistema. Quando a contagem atinge zero, o LED é aceso, proporcionando uma experiência de usuário completa.

Como perspectivas futuras, há a possibilidade de implementar ajustes configuráveis para a contagem inicial, bem como adicionar funcionalidades interativas para aprimorar a versatilidade do contador.

Em resumo, o projeto atingiu seus objetivos ao fornecer um contador regressivo robusto e funcional, utilizando VHDL e máquinas de estados. A integração eficiente de componentes e funcionalidades adicionais contribui para um sistema completo, abrindo caminho para futuras explorações na área de sistemas embarcados e design de circuitos digitais. Este trabalho representa uma base sólida para futuros desenvolvimentos e aprimoramentos.