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

Universidade Tecnológica Federal do Paraná

Engenharia de Computação

PROJETO UTILIZANDO CÓDIGO SEQUENCIAL

Maria Eduarda Pedroso

Professor orientador: Marcelo de Oliveira

Apucarana

Outubro/2023

SUMÁRIO

1. RESUMO..... 2
2. INTRODUÇÃO..... 3
2.1. Objetivos..... 4
3. METODOLOGIA..... 5
3.1. Funcionalidades do projeto..... 5
3.2. Código VHDL..... 7
3.3. PinPlanner..... 9
4. RESULTADOS E DISCUSSÃO..... 10
5. CONSIDERAÇÕES FINAIS..... 14

1. RESUMO

No presente trabalho, exploramos a implementação de um circuito lógico em uma FPGA, com auxílio do software Quartus. O projeto é centrado no desenvolvimento de um circuito, utilizando a linguagem VHDL, que controla uma sequência de LEDs, acendendo-os de forma sequencial do LED0 ao LED9 e retornando do LED9 ao LED0. Diversas funcionalidades auxiliares foram integradas, incluindo a habilidade de resetar e pausar o circuito através de chaves e botões. Em um nível mais profundo, a atividade proporcionou uma compreensão robusta da programação sequencial, fundamental para a engenharia de sistemas digitais. O relatório a seguir detalha as etapas de desenvolvimento, implementação, simulação e resultados alcançados.

Palavras-chave: FPGA, Quartus, Lógica Reconfigurável, Programação Sequencial, LEDs.

2. INTRODUÇÃO

No cenário atual da eletrônica digital, a programação sequencial emerge como uma das ferramentas vitais para a concepção de sistemas que operam com base em sequências de eventos ou estados. Essencialmente, a programação sequencial difere da programação combinacional em que, no primeiro, a saída depende não apenas das entradas atuais, mas também dos estados anteriores do sistema. A linguagem VHDL, uma das principais linguagens de descrição de hardware, provê um conjunto robusto de ferramentas e estruturas para representar esses sistemas sequenciais com precisão e eficiência.

As FPGAs (Field-Programmable Gate Arrays) surgem como palcos ideais para a implementação de tais sistemas descritos em VHDL. Caracterizadas por sua reconfigurabilidade, as FPGAs permitem que os designers criem, testem e modifiquem suas soluções digitais em um ambiente altamente flexível. A capacidade de uma FPGA de ser reprogramada para realizar qualquer operação lógica a torna uma escolha preferida para prototipagem rápida e desenvolvimento de sistemas digitais.

Dentro deste contexto, o software Quartus, da Intel, estabelece-se como uma das plataformas líderes em design e simulação para FPGAs. Proporcionando uma interface intuitiva juntamente com ferramentas de análise avançada, o Quartus facilita a transição do código VHDL para uma implementação funcional na FPGA.

O intuito desta atividade é consolidar a compreensão teórica de programação sequencial em VHDL através de uma aplicação prática: a criação de um circuito lógico que controla uma sequência de LEDs. Além do acendimento sequencial dos LEDs, o projeto também engloba funcionalidades auxiliares como resetar e pausar o

circuito, proporcionando uma experiência abrangente no desenvolvimento de sistemas sequenciais em FPGAs com o auxílio do Quartus. Ao final desta jornada, espera-se que os participantes não apenas tenham fortalecido suas habilidades em VHDL e design de FPGA, mas também compreendam profundamente os princípios subjacentes da programação sequencial e sua relevância no mundo da eletrônica digital.

2.1. Objetivos

O presente trabalho tem como objetivo:

- Entendimento Profundo de Programação Sequencial: Adquirir uma compreensão robusta da programação sequencial, compreendendo como a saída de um sistema pode depender tanto das entradas atuais quanto de estados anteriores.
- Aplicação Prática em VHDL: Utilizar a linguagem de descrição de hardware VHDL para projetar e simular um circuito sequencial, solidificando a teoria através de aplicações práticas.
- Familiarização com FPGA: Ganhar experiência no uso de FPGAs, compreendendo suas características reconfiguráveis e a importância destes dispositivos no cenário atual de design de sistemas digitais.
- Proficiência no Software Quartus: Aperfeiçoar habilidades no software Quartus, aprendendo a programar eficazmente de um design de VHDL para uma implementação funcional na FPGA.
- Design de Circuitos Lógicos: Desenvolver um circuito que controla o acendimento sequencial de LEDs, integrando funcionalidades de reset e pausa, e assim aprimorar habilidades de design e troubleshooting em sistemas digitais.

- Integração de Componentes de Interface: Integrar e configurar componentes físicos, como chaves e botões, para interagir com o circuito lógico, proporcionando uma experiência completa de design de sistemas interativos.
- Análise e Refinamento: Após a implementação, realizar simulações e testes práticos para analisar o comportamento do circuito, identificando possíveis áreas de melhoria e refinando o design conforme necessário.

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 implementar na placa nossa lógica.

3.1. Funcionalidades do projeto

3.1.1. Contador e LEDs

Para a lógica dos leds utilizamos os clocks para conseguir um tempo uniforme e correto e dentro dessa transição do clock foi utilizado um contador para saber qual led deveria ser aceso no momento, incrementamos o contador até ele ser menor que o tempo piscar e quando ele é maior rejeitamos o contador e atualizamos o array de leds para a próxima posição. Abaixo temos a simplificação desse trecho do código.

```
constant tempo_piscar: integer := f_clk/2;
```

```
elsif rising_edge(clk) then --TRANSIÇÃO POSITIVA DO CLOCK (clock subiu)
                        if pause /= '1' then
```

```
                    if counter < tempo_piscar then
        counter := counter + 1;
                    else
                        counter := 0;

                    end if;
```

3.1.2. Lógica de Controle Leds

Como nosso array de leds tem 10 posições e precisamos ir e voltar usamos uma lógica de direções, se o lugar do led é 8 ou seja posição 9 ele muda de direção para voltar o array, caso o contrário ele segue o array de forma normal incrementando os led, foi utilizado um switch case para que o vetor fique apenas com a posição necessária em alta, poderia ter usado um shift lógico mas a fim de facilitar a implementação optou-se pelo switch case.

```
--MUDANÇA DAS DIREÇOES
        if lugar_led = 8 then
                dir <= 0;  -- Mudar direção
        elsif lugar_led = 1 then
                dir <= 1;   -- Mudar direção
        end if;
```

3.1.3. Reset do Circuito

Para resetar o circuito a lógica é simples, quando apontamos o botão que simboliza o reset ele fica em baixa e entra no if abaixo, dentro desse if apagamos todos os Leds para que fique mais agradável visualmente, e voltamos os valores de contador, direção para o inicial.

```
--QUANDO RESETAMOS O TREM
        if rst = '0' then
                counter := 0;
                leds <= "0000000000";
                dir <= 1;
```

```
lugar_led <= 0;
```

3.1.4. Pausa do Circuito

Para pausar o circuito utilizamos um swich, basicamente quando o switch está em alta apenas desligamos todos os leds e as outras variaveis ficam estagnadas, esperando para quando despausar retornar ao estado no qual pausamos.

```
if pause /= '1' then
    - CODIGO FUNCIONAL
else
    leds <= "0000000000";
end if;
```

3.2. Código VHDL

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

```
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.NUMERIC_STD.ALL;
--TODO: ARRUMAR CLOCK
----------------------------------------------------------------------------------------------------
entity aula230830 is
                    generic(
                    --f_clk : integer := f_clk'high; -- variavel contadora
                    f_clk : integer := 50_000_000
                    );
  Port ( clk   : in STD_LOGIC;  -- usamos clock de 50Mhz PIN_P11
      rst   : in STD_LOGIC;  -- botão de reset
      pause  : in STD_LOGIC;  -- botão de pausa
      leds   : out STD_LOGIC_VECTOR(9 downto 0));
```

```
end aula230830;
-------------------------------------------------------------------------------------------------------
architecture Behavioral of aula230830 is
        --variaveis
        signal lugar_led : integer range 0 to 9 := 0;
  signal dir    : integer := 1;  -- Direção: 1 para avançar, -1 para voltar
        constant tempo_piscar: integer := f_clk/2;
begin

  process(clk, rst)
        variable counter: integer RANGE 0 TO f_clk'high;
  begin

                --QUANDO RESETAMOS O TREM
                if rst = '0' then
                        counter := 0;
                        leds <= "0000000000";
                        dir <= 1;
                        lugar_led <= 0;

                --SE NÃO FOI RESETADO TEMOS QUE VER QUAL DIREÇÃO E MUDAR LED
                elsif rising_edge(clk) then --TRANSIÇÃO POSITIVA DO CLOCK (clock subiu)

                        if pause /= '1' then

                                if counter < tempo_piscar then  -- Queremos mudar o LED a
cada 10 pulsos de clock? 10^50(MILHOES)
                                        counter := counter + 1;
                                else
                                        counter := 0;
                                        case lugar_led is
                                                when 0 => leds <= "0000000001";
                                                when 1 => leds <= "0000000010";
                                                when 2 => leds <= "0000000100";
                                                when 3 => leds <= "0000001000";
                                                when 4 => leds <= "0000010000";
                                                when 5 => leds <= "0000100000";
                                                when 6 => leds <= "0001000000";
                                                when 7 => leds <= "0010000000";
                                                when 8 => leds <= "0100000000";
                                                when 9 => leds <= "1000000000";
                                        end case;

                                        if dir = 1 then
                                                lugar_led <= lugar_led + 1;
                                        elsif dir = 0 then
                                                lugar_led <= lugar_led - 1;
```

```
                                    end if;

                                    --MUDANÇA DAS DIREÇOES
                                    if lugar_led = 8 then
                                        dir <= 0;  -- Mudar direção
                                    elsif lugar_led = 1 then
                                        dir <= 1;  -- Mudar direção
                                    end if;

                                end if;
                            else
                                leds <= "0000000000";
                            end if;
                        end if;
                    end process;

end Behavioral;
```

3.3. PinPlanner

Para esse projeto precisaremos utilizar um botão para resetar, um switch para pausar e os leds, com isso precisamos montar o pinPlanner com as localizações certas, meu pinPlanner está com essa configuração abaixo.

Figura 1 - PinPlaner

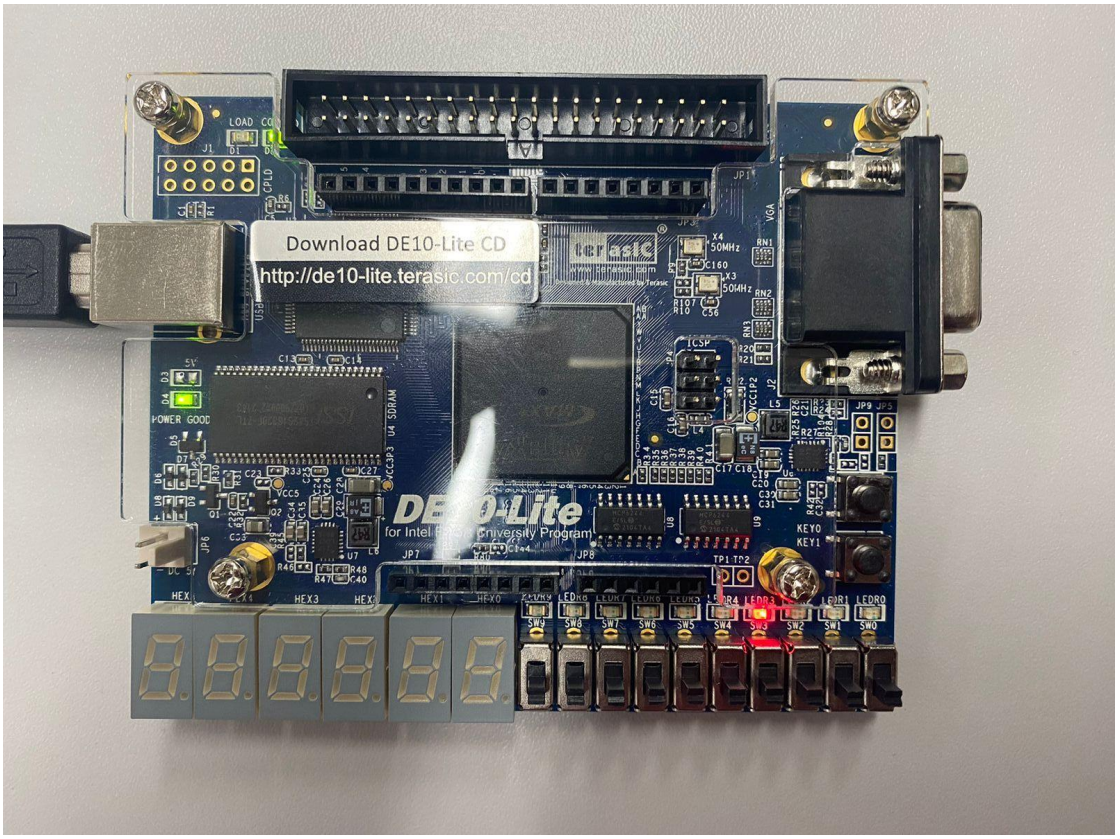
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
clk	Input	PIN_P11	3	B3_N0	PIN_P11	2.5 V		12mA (default)	
leds[9]	Output	PIN_B11	7	B7_N0	PIN_B11	2.5 V		12mA (default)	2 (default)
leds[8]	Output	PIN_A11	7	B7_N0	PIN_A11	2.5 V		12mA (default)	2 (default)
leds[7]	Output	PIN_D14	7	B7_N0	PIN_D14	2.5 V		12mA (default)	2 (default)
leds[6]	Output	PIN_E14	7	B7_N0	PIN_E14	2.5 V		12mA (default)	2 (default)
leds[5]	Output	PIN_C13	7	B7_N0	PIN_C13	2.5 V		12mA (default)	2 (default)
leds[4]	Output	PIN_D13	7	B7_N0	PIN_D13	2.5 V		12mA (default)	2 (default)
leds[3]	Output	PIN_B10	7	B7_N0	PIN_B10	2.5 V		12mA (default)	2 (default)
leds[2]	Output	PIN_A10	7	B7_N0	PIN_A10	2.5 V		12mA (default)	2 (default)
leds[1]	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V		12mA (default)	2 (default)
leds[0]	Output	PIN_A8	7	B7_N0	PIN_A8	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)	

Fonte: Autoria Própria

4. RESULTADOS E DISCUSSÃO

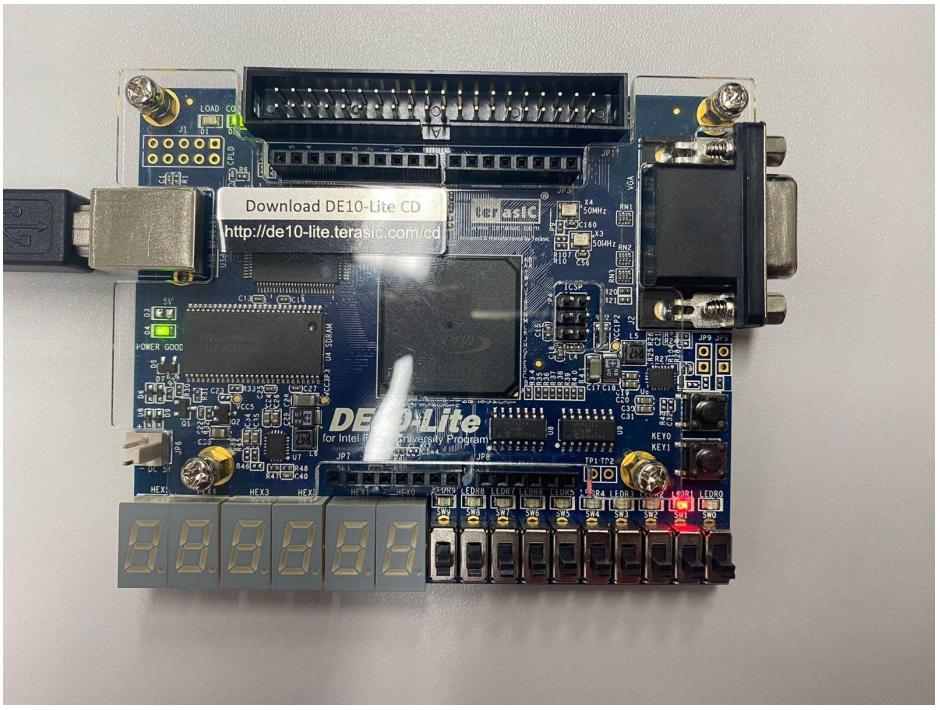
Com a implementação e pinPlaner prontos foi startado o código na placa, os resultados foram condizentes com a entrada como podemos visualizar nessa imagem abaixo

Figura 2 - Estado do led indo para a esquerda



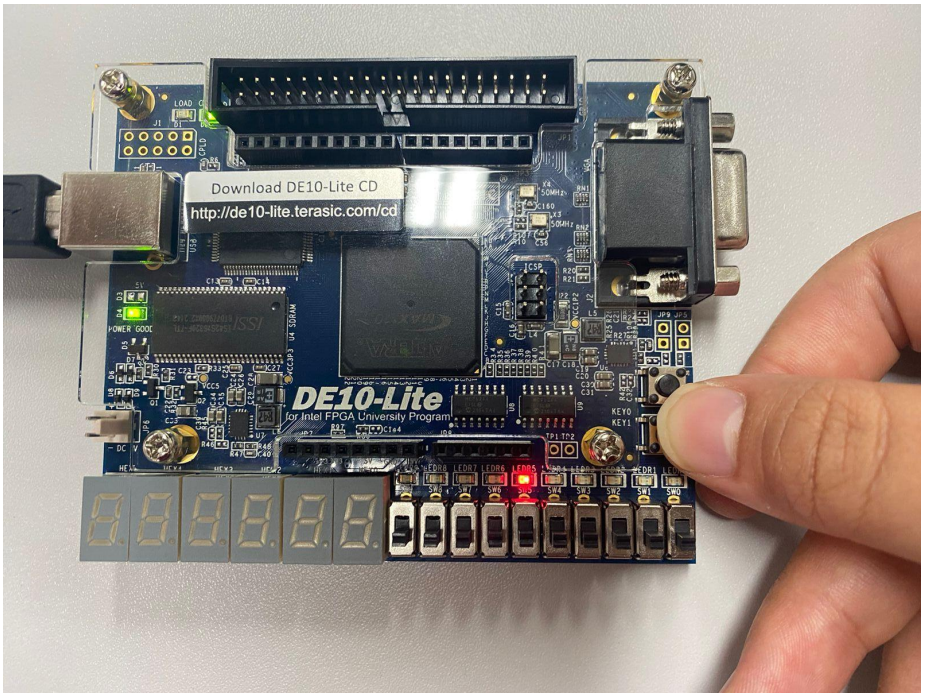
Fonte: Autoria Própria

Figura 3 - Estado do Led indo para a direita



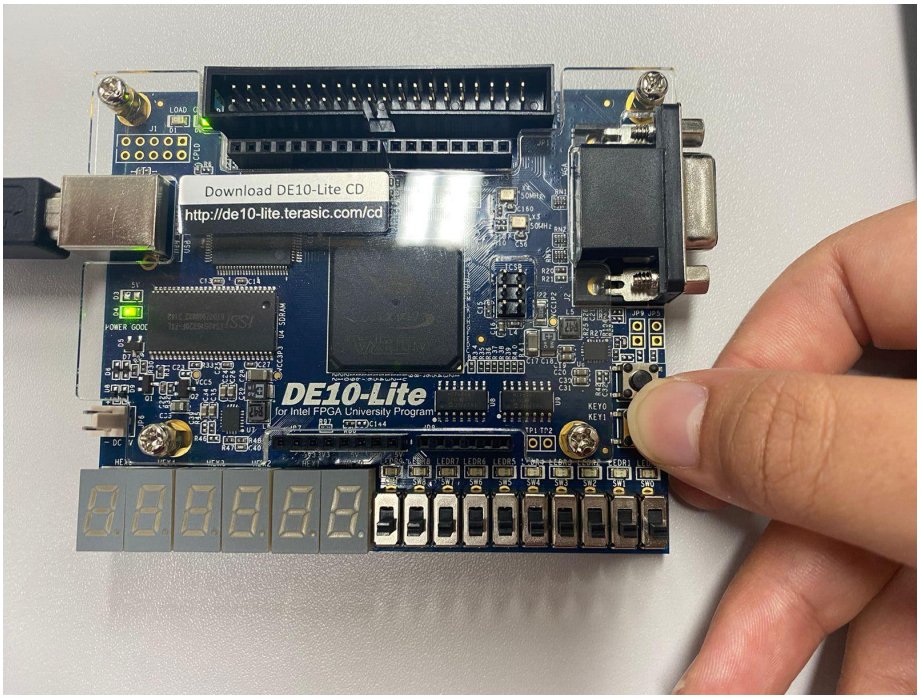
Fonte: Autoria Própria

Figura 4 - Estado antes de pressionar o reset



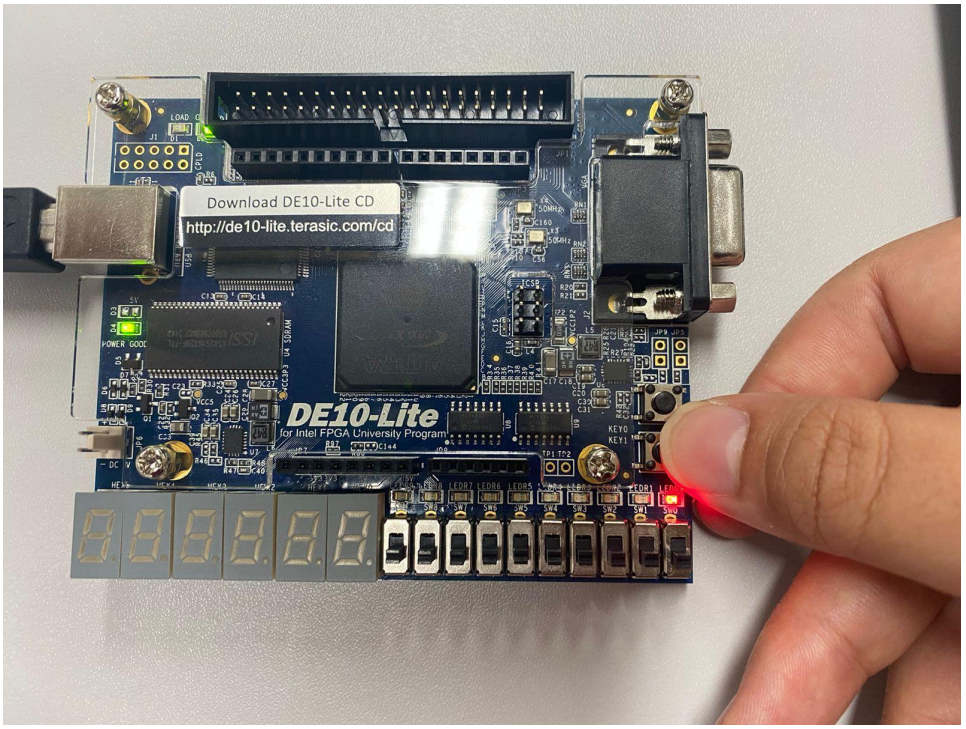
Fonte: Autoria Própria

Figura 5 - Reset pressionado



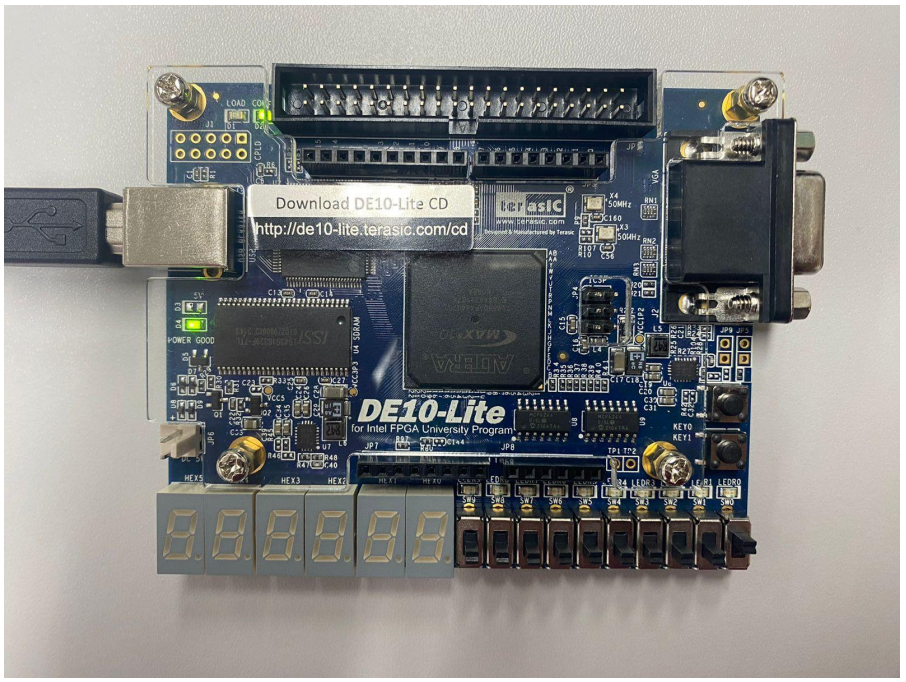
Fonte: Autoria Própria

Figura 6 - Reset funcionando



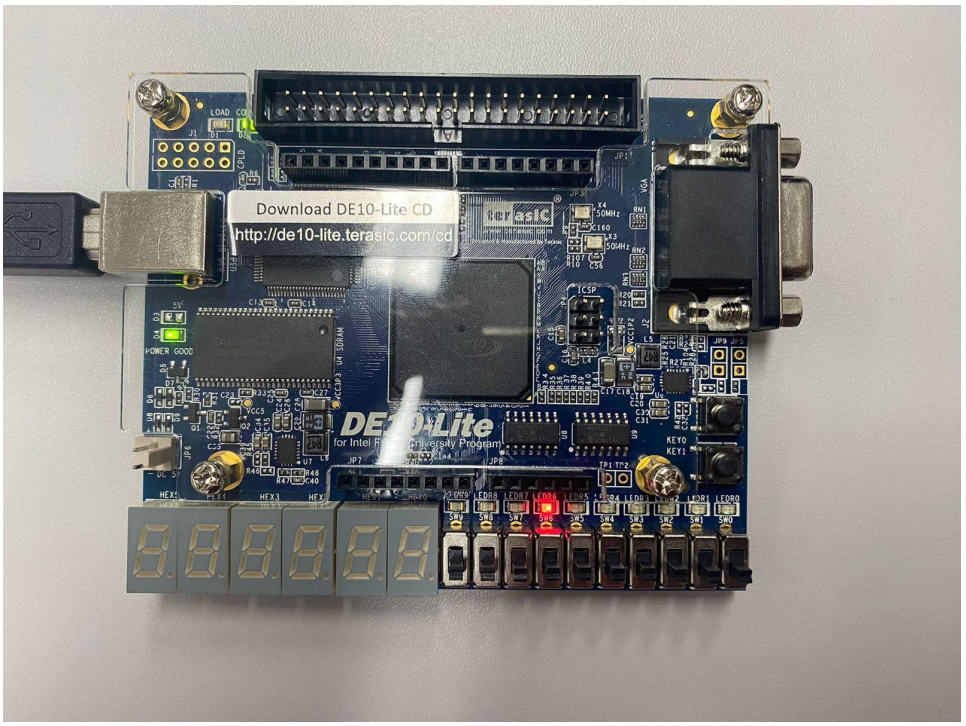
Fonte: Autoria Própria

Figura 7 - Pause ligado



Fonte: Autoria Própria

Figura 8 - Pause desligado



Fonte: Autoria Própria

5. CONSIDERAÇÕES FINAIS

Ao concluirmos esta atividade prática de lógica reconfigurável com FPGA, é possível refletir sobre a importância e a relevância dos conceitos e técnicas aprendidos no panorama atual da eletrônica digital. A programação sequencial, sendo um pilar central do design de sistemas digitais, foi não apenas discutida teoricamente, mas vivenciada de forma prática, permitindo uma assimilação mais profunda do conteúdo.

A utilização da linguagem VHDL para descrever e simular nosso circuito proporcionou uma experiência hands-on valiosa. Através deste exercício, observou-se o poder e a versatilidade de VHDL como ferramenta de descrição de hardware, especialmente quando se trata de sistemas sequenciais.

A plataforma FPGA, com sua natureza reconfigurável, demonstrou ser um ambiente ideal para prototipagem e testes. A capacidade de programar, reprogramar e ajustar nosso design em tempo real permitiu um ciclo de feedback rápido e eficiente, crucial para o refinamento e aperfeiçoamento do projeto.

O software Quartus, por sua vez, mostrou-se indispensável ao facilitar a transição entre o código VHDL e a implementação real na FPGA. Através de suas ferramentas e interfaces intuitivas, pudemos simular, analisar e depurar nosso projeto com eficácia.

O desafio de criar um circuito que controlasse uma sequência de LEDs, integrando funcionalidades como reset e pausa, revelou-se não apenas instrutivo, mas também inspirador. Este projeto, apesar de sua aparente simplicidade, encapsulou diversos conceitos fundamentais da eletrônica digital e serviu como um lembrete de como os princípios básicos podem ser aplicados de formas variadas e criativas.

Por fim, esta atividade reforçou a ideia de que a aprendizagem é mais eficaz quando teoria e prática caminham juntas. O equilíbrio entre conceitos teóricos, simulações e implementações práticas proporcionou uma compreensão holística e aprofundada da programação sequencial e suas aplicações.

Em retrospecto, a jornada foi enriquecedora e instigante, e esperamos aplicar as lições aprendidas em futuros projetos e desafios na área da eletrônica digital.