

RELATÓRIO DE SERVIÇO

SOLICITANTE – Setor de Engenharia da Uninter

TÍTULO DO PROJETO: Migração de Circuito Discreto para Implementação em FPGA

DATA DE ENTREGA: 07 de abril de 2025

ENGENHEIRO RESPONSÁVEL: Eder Naloto Machado

RU/CREA DO ENGENHEIRO: 3881863

1. DESCRIÇÃO DO PROJETO (1 ponto)

Objetivo Principal:

- Desenvolver um procedimento de engenharia reversa para um circuito específico, originalmente projetado na plataforma Tinkercad, com o objetivo de transferir sua funcionalidade para um código VHDL, para implementação em um dispositivo FPGA.

Características do Circuito:

- O circuito possui três entradas binárias controladas por interruptores.
- O circuito possui quatro saídas discretas representadas por LEDs (adicionados apenas para visualização).

Metodologia:

- Foi realizada uma simulação detalhada no ambiente Tinkercad para obter as informações necessárias para o desenvolvimento do código VHDL.
- A partir da simulação, os dados coletados foram utilizados como base para a criação do código VHDL.

Resultados e Análise:

- Durante o processo de simulação, foram organizadas e apresentadas as conclusões em tabelas.
- As contagens foram feitas considerando o LED4 como o bit mais significativo (MSB).

Ativa	Atualiza	Operação	LEDs
1	1 → 0	0	Contagem crescente
0	Qualquer	0	0
Qualquer	1 → 0	1	Contagem decrescente

2. ESPECIFICAÇÕES DO SOFTWARE UTILIZADO

Software Utilizado: Quartus II

- **Desenvolvedor:** ALTERA
- **Descrição:** O Quartus II é um software de design especializado em soluções para lógica programável, desenvolvido pela ALTERA.
- **Tipo:** Ambiente de Desenvolvimento Integrado (IDE - Integrated Development Environment).
- **Função:** Utilizado para o desenvolvimento de projetos com componentes ALTERA, permitindo a programação e implementação em dispositivos FPGA.
- **Capacidade:** O Quartus II suporta a compilação tanto de esquemáticos quanto de código em VHDL para programação de FPGAs.

3. CÓDIGO DESENVOLVIDO EM VHDL E DIAGRAMA RTL (1 ponto)

Figura 1 – Código em VHDL

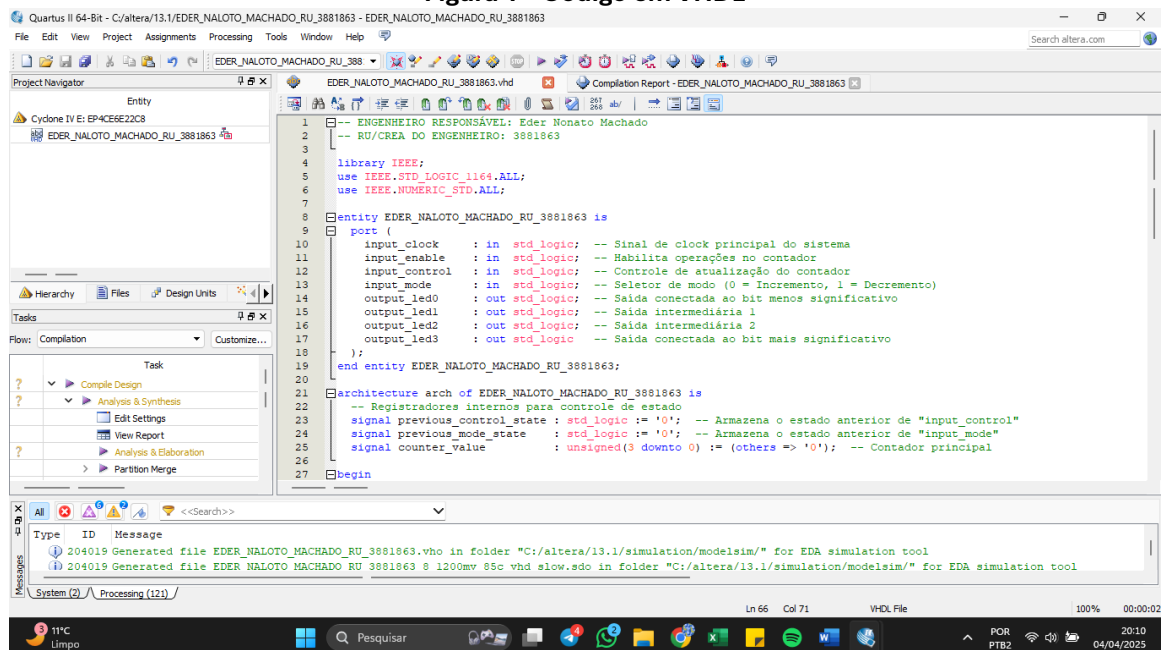
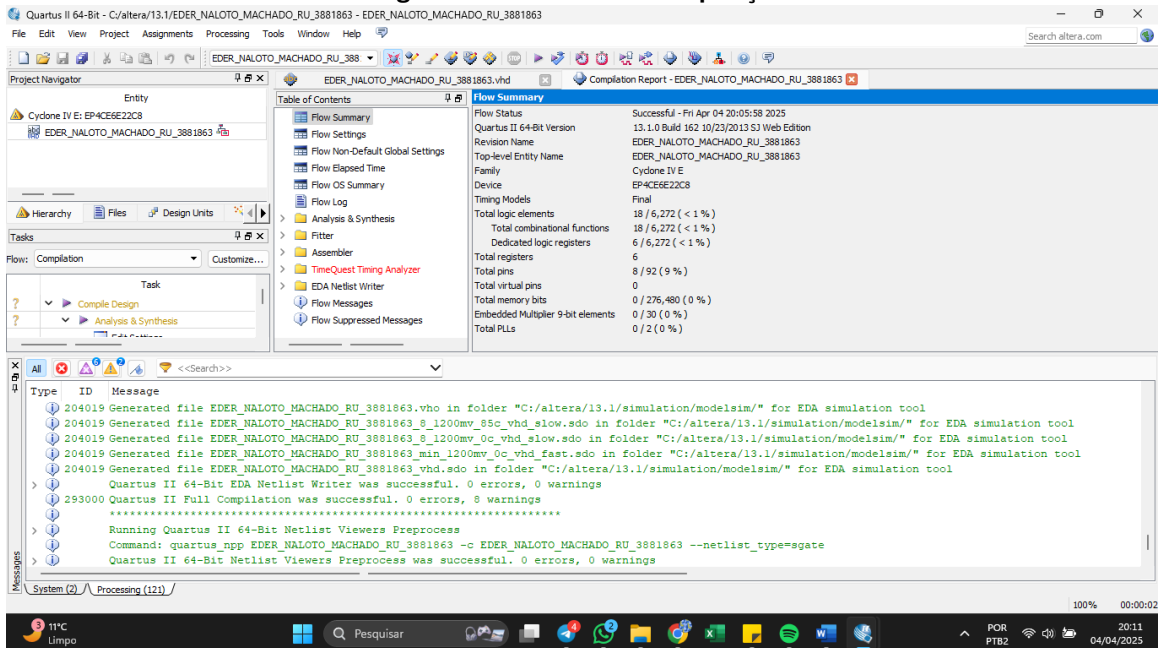


Figura 2 – Relatório de compilação



The screenshot displays the Quartus II 64-bit compilation report for the project 'EDER_NALOTO_MACHADO_RU_3881863'. The 'Flow Summary' tab is active, showing the following statistics:

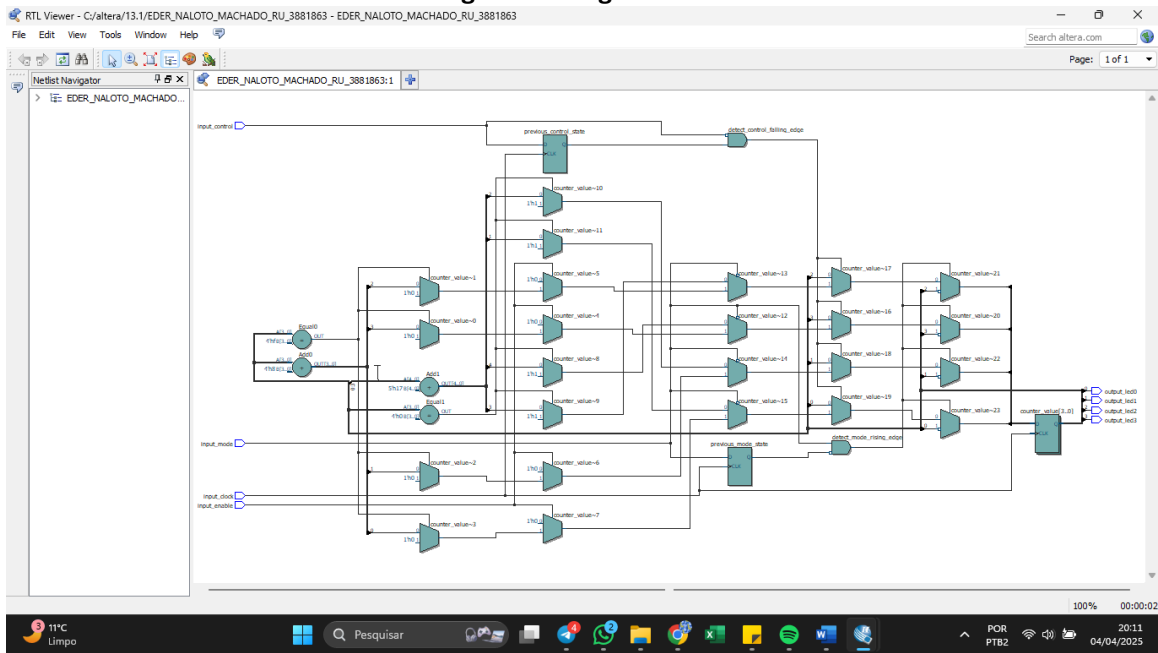
Flow Status	Successful - Fri Apr 04 20:05:58 2025
Quartus II 64-bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	EDER_NALOTO_MACHADO_RU_3881863
Top-level Entity Name	EDER_NALOTO_MACHADO_RU_3881863
Family	Cyclone IV E
Device	EP4CE6E22C8
Timing Models	Final
Total logic elements	18 / 6,272 (< 1 %)
Total combinational functions	18 / 6,272 (< 1 %)
Dedicated logic registers	6 / 6,272 (< 1 %)
Total registers	6
Total pins	8 / 92 (9 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 20 (0 %)
Total PLLs	0 / 2 (0 %)

The 'Messages' window at the bottom shows the following messages:

```

204019 Generated file EDER_NALOTO_MACHADO_RU_3881863.vho in folder "C:/altera/13.1/simulation/modelsim/" for EDA simulation tool
204019 Generated file EDER_NALOTO_MACHADO_RU_3881863_0_1200mv_85c_vhd_slow.sdo in folder "C:/altera/13.1/simulation/modelsim/" for EDA simulation tool
204019 Generated file EDER_NALOTO_MACHADO_RU_3881863_0_1200mv_0c_vhd_slow.sdo in folder "C:/altera/13.1/simulation/modelsim/" for EDA simulation tool
204019 Generated file EDER_NALOTO_MACHADO_RU_3881863_min_1200mv_0c_vhd_fast.sdo in folder "C:/altera/13.1/simulation/modelsim/" for EDA simulation tool
204019 Generated file EDER_NALOTO_MACHADO_RU_3881863.vhd.sdo in folder "C:/altera/13.1/simulation/modelsim/" for EDA simulation tool
> Quartus II 64-bit EDA Netlist Writer was successful. 0 errors, 0 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 0 warnings
> Running Quartus II 64-bit Netlist Viewers Preprocess
Command: quartus_npp EDER_NALOTO_MACHADO_RU_3881863 -c EDER_NALOTO_MACHADO_RU_3881863 --netlist_type=gate
> Quartus II 64-bit Netlist Viewers Preprocess was successful. 0 errors, 0 warnings
    
```

Figura 3 – Diagrama RTL



4. SIMULAÇÃO DO WAVEFORM (3 pontos)

Objetivo da Simulação:

- Os resultados da simulação foram apresentados por meio da forma de onda (waveform), conforme mostrado na Figura 4.
- A simulação iniciou-se a partir de 404 ns, marcado como o início do RU (Registro de Unidade).

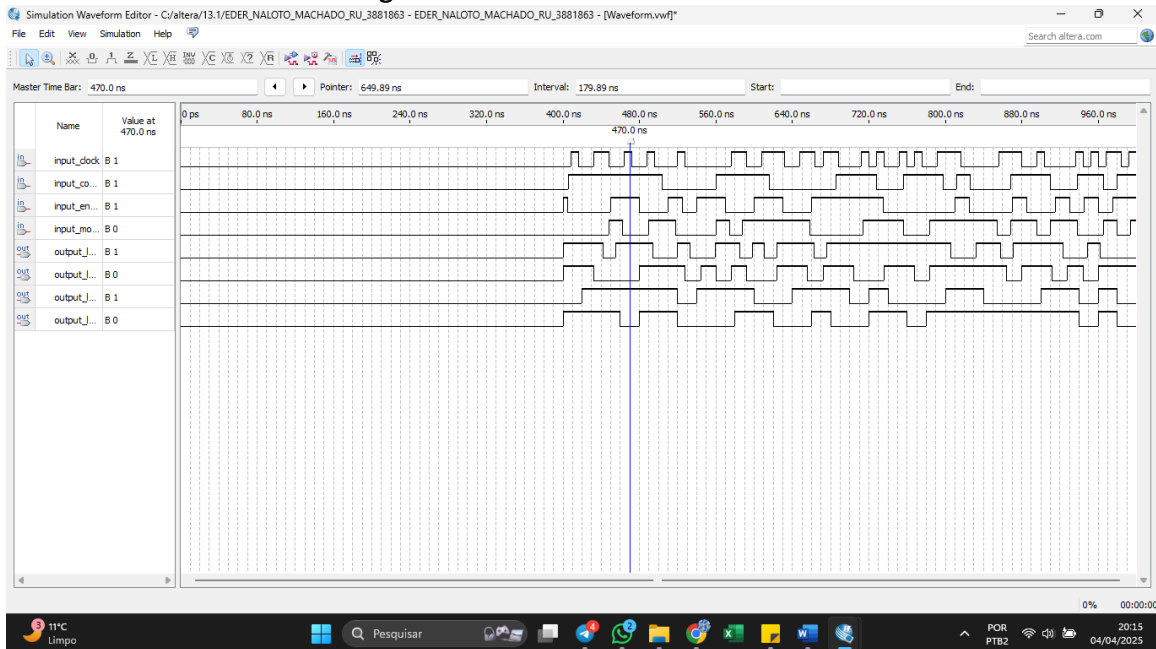
Configurações da Simulação:

- Clock de Entrada: Um clock de 1 ns foi utilizado como entrada.
- Entradas: Foram utilizadas as três entradas binárias (INT 1, INT 2, e INT 3) conforme o circuito original.
- Variação das Entradas: Foram testadas as 8 possibilidades de entrada (000, 001, 010, 011, 110, 100, 101, 111) com diferentes períodos de clock nas entradas INT 1, 2 e 3.

Observações durante os Intervalos:

- I. Primeiro Intervalo:
 - Quando INT3 está em nível alto, os LEDs começam a contar de forma decrescente.
 - A contagem decrescente ocorre devido à mudança do nível de INT2 de alto para baixo, mesmo quando INT1 está em nível alto.
- II. Segundo Intervalo:
 - Quando INT1 e INT3 estão em nível baixo, todos os LEDs ficam em nível baixo (apagados).
- III. Terceiro Intervalo:
 - Quando INT1 está em nível alto, a contagem inicia de forma crescente.
- IV. Quarto Intervalo:
 - Quando INT3 sobe de nível baixo para alto, a saída do LED se inverte binariamente.
 - Por exemplo, a contagem muda de 0111 (7) para 1000 (8) e continua com a contagem decrescente.

Figura 4 – Waveform com resultados



6. DISCUSSÃO DOS RESULTADOS (3 pontos)

Fidelidade dos Resultados:

- Os resultados da simulação mostraram uma correspondência precisa com o comportamento do circuito original.
- A abordagem adotada foi eficaz, comprovando a confiança na metodologia utilizada.

Simplicidade na Implementação:

- A utilização de LEDs como indicador simplificou significativamente a codificação.
- A possibilidade de realizar operações de soma binária em VHDL eliminou a necessidade de definir manualmente cada estado sequencial do contador, tornando o processo mais eficiente.

Desafio na Identificação da Lógica de Acendimento dos LEDs:

- A identificação da lógica de acionamento dos LEDs foi um desafio complexo devido à presença de múltiplas combinações possíveis e lógica sequencial no circuito.
- A lógica sequencial estava vinculada a transições do BIT 2 (para baixo) e à inversão da contagem com a transição do BIT 3 (para cima).
- Superar esse desafio exigiu um processo iterativo de testes e análises até que o funcionamento do circuito fosse completamente compreendido.

Oportunidades de Aprimoramento:

I. Escalabilidade:

- Em circuitos maiores, com muitas entradas e saídas, a abordagem de engenharia reversa por tentativa e erro na simulação teria limitações.
- Para circuitos mais complexos, seria mais eficaz elaborar documentação detalhada do circuito original, o que facilitaria a implementação em outras tecnologias, como FPGAs.

II. Substituição da Lógica de Contagem:

- A substituição da lógica de contagem dos LEDs por uma implementação sem o uso de registradores/contadores poderia ser uma estratégia útil.
- Essa mudança permitiria uma análise comparativa do consumo de recursos e desempenho em placas FPGA, possibilitando otimizações no projeto.

7. FORMATAÇÃO E FORMATO DE ENTREGA DE TODO O PROJETO (2)

O projeto está sendo entregue no formato

projeto_final_EDER_NALOTO_MACHADO_RU_3881863.zip

OBSERVAÇÕES:

ASSINATURA DO ENGENHEIRO RESPONSÁVEL:

_____ **Eder Naloto Machado** _____

Engenheiro Eletricista

RU: 3881863

UNINTER ENGENHARIA.

Código em VHD

--SOLICITANTE – Setor de Engenharia da Uninter

--TÍTULO DO PROJETO: Migração de Circuito Discreto para Implementação em FPGA

--DATA DE ENTREGA: 07 de abril de 2025

-- ENGENHEIRO RESPONSÁVEL: Eder Nonato Machado

-- RU/CREA DO ENGENHEIRO: 3881863

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

entity EDER_NALOTO_MACHADO_RU_3881863 is

port (

input_clock : in std_logic; -- Sinal de clock principal do sistema

input_enable : in std_logic; -- Habilita operações no contador

input_control : in std_logic; -- Controle de atualização do contador

input_mode : in std_logic; -- Seletor de modo (0 = Incremento, 1 = Decremento)

output_led0 : out std_logic; -- Saída conectada ao bit menos significativo

output_led1 : out std_logic; -- Saída intermediária 1

output_led2 : out std_logic; -- Saída intermediária 2

output_led3 : out std_logic -- Saída conectada ao bit mais significativo

);

end entity EDER_NALOTO_MACHADO_RU_3881863;

architecture arch of EDER_NALOTO_MACHADO_RU_3881863 is

-- Registradores internos para controle de estado

signal previous_control_state : std_logic := '0'; -- Armazena o estado anterior de
"input_control"

signal previous_mode_state : std_logic := '0'; -- Armazena o estado anterior de
"input_mode"

signal counter_value : unsigned(3 downto 0) := (others => '0'); -- Contador principal

begin

-- Processo Sequencial Síncrono

process(input_clock)

variable detect_mode_rising_edge : boolean := false; -- Detecta borda de subida no
seletor de modo

variable detect_control_falling_edge : boolean := false; -- Detecta borda de descida no
sinal de controle

begin

if rising_edge(input_clock) then

-- Atualiza os estados anteriores das entradas

previous_control_state <= input_control;

previous_mode_state <= input_mode;

-- Detecta borda de subida no seletor de modo

detect_mode_rising_edge := (previous_mode_state = '0' and input_mode = '1');

-- Detecta borda de descida no sinal de controle

detect_control_falling_edge := (previous_control_state = '1' and input_control = '0');


```

-- Executa a lógica com base nas detecções

if detect_mode_rising_edge then

    counter_value <= not counter_value; -- Inverte todos os bits do contador


elsif detect_control_falling_edge then

    -- Lógica de operação baseada no modo selecionado

    if input_mode = '0' then -- MODO INCREMENTAL

        if input_enable = '1' then

            if counter_value = "1111" then

                counter_value <= "0000"; -- Reinicia em caso de overflow

            else

                counter_value <= counter_value + 1; -- Incrementa o contador

            end if;

        else

            counter_value <= "0000"; -- Zera o contador quando desabilitado

        end if;

    else -- MODO DECREMENTAL

        if counter_value = "0000" then

            counter_value <= "1111"; -- Satura em caso de underflow

        else

            counter_value <= counter_value - 1; -- Decrementa o contador

        end if;

    end if;

end if;

```

```
    end if;

    end if;

end process;


-- Conexão direta dos bits do contador às saídas de LED

output_led0 <= counter_value(0); -- Bit menos significativo

output_led1 <= counter_value(1);

output_led2 <= counter_value(2);

output_led3 <= counter_value(3); -- Bit mais significativo


end architecture arch;
```