

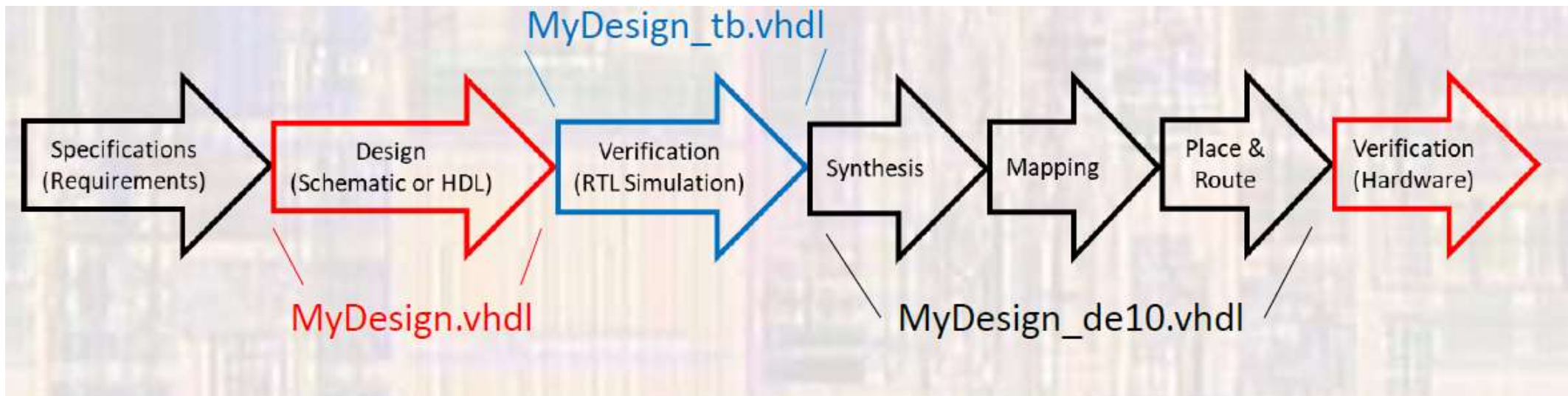
Primeiro Projeto em VHDL

Jorge Amaral

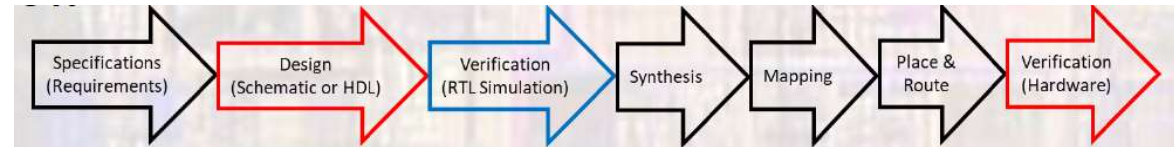
Exemplo de um contador

- Criar um projeto lógico do início até a implementação em uma placa DE10
- Hierarquia de projeto
 - Nomeie seu testbench com o mesmo nome do seu arquivo de design, mas adicione _tb no final
 - counter_8bit.vhdl -> counter_8bit_tb.vhdl
 - Quando começarmos a usar o DE10 faremos a mesma coisa
 - counter_8bit.vhdl -> counter_8bit_de10.vhdl
- Em ambos os casos, nunca alteramos o arquivo de design base

Fluxo de Projeto

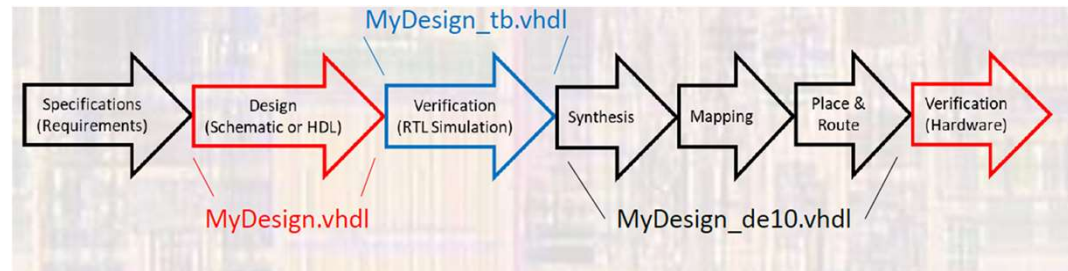


Fluxo



- Criar projeto
- Criar código HDL
- Verificar RTL visualmente
- Criar testbench
- Verificar operação
- Criar implementação DE10 HDL
- Compilar
- Verificar RTL visualmente
- Programar placa DE10
- Verificar operação manualmente

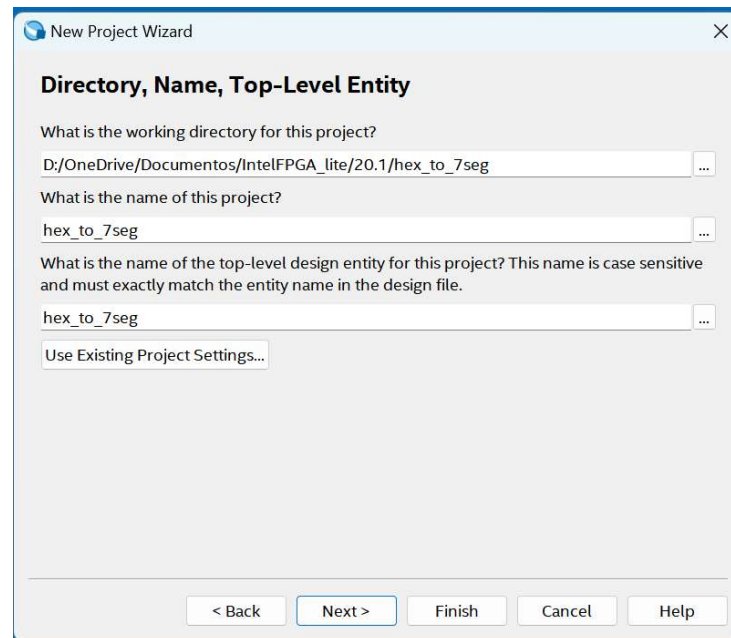
Fluxo



- Crie um contador
- Contagem binária de N bits
- Para cima/para baixo
- Teste o design usando o ModelSim
- Implemente uma versão de 8 bits no DE10
- Reinicialização e direção através das chaves
- Valor da contagem nos LEDs
- Divisor de clock para a operação em 3 Hz

Criar um novo projeto no Quartus

- File -> New Project Wizard
- Use as notas de setup de projeto



The screenshot shows the 'New Project Wizard' dialog box with the title bar 'New Project Wizard' and a close button. The main section is titled 'Directory, Name, Top-Level Entity'. It contains three text input fields, each with a browse button (three dots) to its right. The first field is labeled 'What is the working directory for this project?' and contains the path 'D:/OneDrive/Documentos/IntelFPGA_lite/20.1/hex_to_7seg'. The second field is labeled 'What is the name of this project?' and contains 'hex_to_7seg'. The third field is labeled 'What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.' and also contains 'hex_to_7seg'. Below these fields is a button labeled 'Use Existing Project Settings...'. At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Next >' button is highlighted with a blue border.

- Family: Max 10 (DA/DF/DC/SA/SC)
- Device: Max 10 DA
- Name Filter: 10M50DAF484C7G

New Project Wizard

Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.
To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: MAX 10 DA

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter: 10M50DAF484C7G

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit ele
10M50DAF484C7G	1.2V	49760	360	360	1677312	288

< Back Next > Finish Cancel Help

- Simulation: ModelSim-Altera
- Format: VHDL

EDA Tool Settings

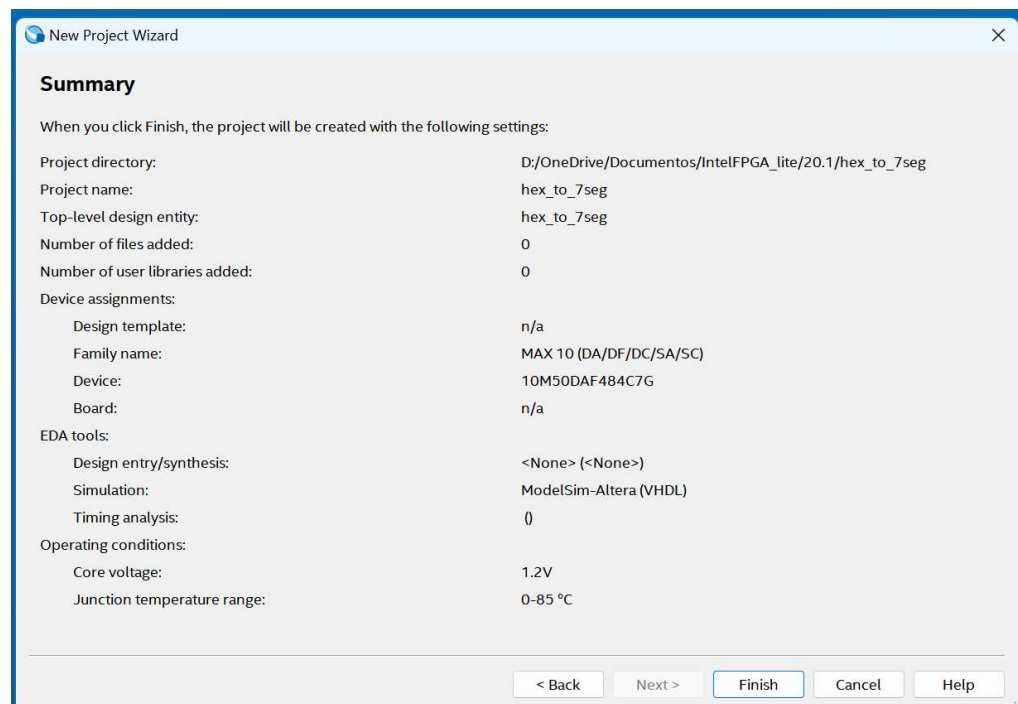
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entr...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	VHDL	<input checked="" type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back **Next >** Finish Cancel Help

- Summary - > Finish



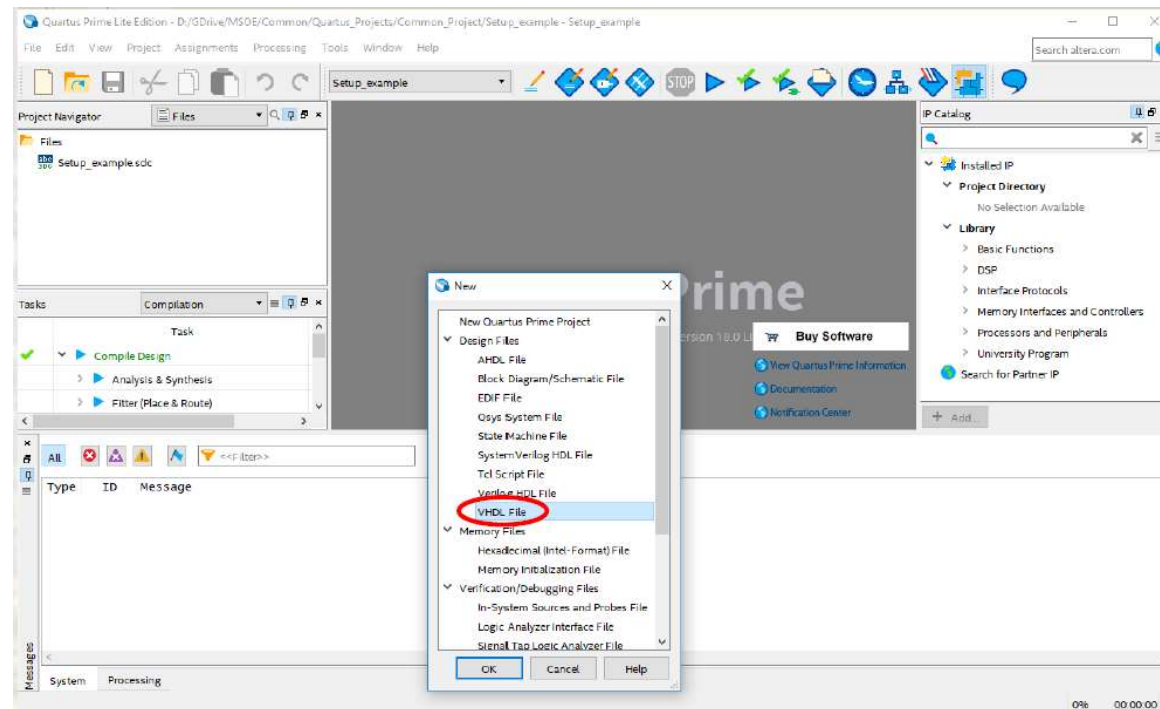
- Usar versão 2008 do VHDL
- Assignments --> Settings --> Compiler Settings --> VHDL Input
- Select VHDL 2008

- Se você planeja executar o projeto no DE10 e você deseja usar os nomes de pinos atribuídos
- Precisa importar o arquivo DE10_Lite.qsf
- Assignments → Imports Assignments

- O posicionamento e o roteamento adequados exigem que o projeto atenda a um conjunto de requisitos de tempo
- Um conjunto muito básico de requisitos de tempo está disponível no arquivo Basic_SDC.sdc
- Você usará isso como um ponto de partida para cada projeto que criar
- Tools → Time Analyzer
 - File → New SDC File
 - Copia e cola do Basic_SDC.sdc
 - File → Save As

Cria um novo arquivo VHDL

- File _> New_ VHDL File

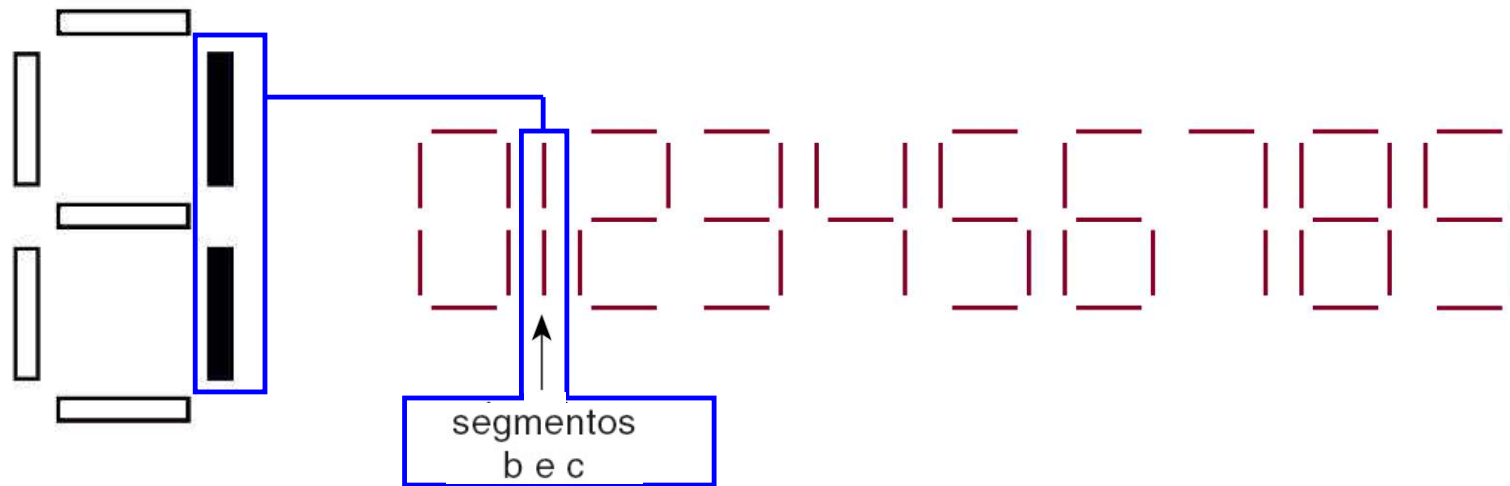


Decodificadores

Decodificadores/ Drivers BCD para 7 segmentos

- O display de 7 segmentos é uma forma usual para mostrar caracteres decimais e hexadecimais.

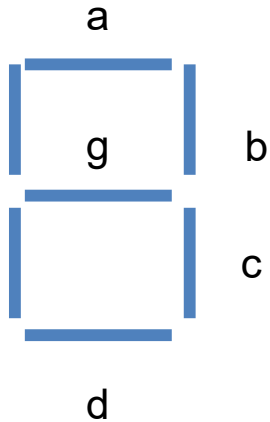
Uma das formas comuns de arranjo utiliza diodos emissores de luz (LEDs) para cada segmento.



Controlando a corrente que passa por cada LED, alguns segmentos são ligados e emitem luz, enquanto outros são desligados, o que gera o padrão do caractere desejado.

Decodificador Hexadecimal para 7 segmentos

De c	Hex	Bin	a	b	c	d	e	f	g
0	0	0000	1	1	1	1	1	1	0
1	1	0001	0	1	1	0	0	0	0
2	2	0010	1	1	0	1	1	0	1
3	3	0011	1	1	1	1	0	0	1
4	4	0100	0	1	1	0	0	1	1
5	5	0101	1	1	0	1	1	0	1
6	6	0110	1	0	1	1	1	1	1
7	7	0111	1	1	1	0	0	0	0
8	8	1000	1	1	1	1	1	1	1
9	9	1001	1	1	1	1	0	1	1
10	A	1010	1	1	1	0	1	1	1
11	B	1011	0	0	1	1	1	1	0
12	C	1100	1	0	0	1	1	1	0
13	D	1101	0	1	1	1	1	0	1
14	E	1110	1	0	0	1	1	1	1
15	F	1111	1	0	0	0	1	1	1



Escreva o decodificador em VHDL

```
library ieee;
use ieee.std_logic_1164.all;

entity hex_to_7seg is
  port (
    i_digit:    in    std_logic_vector(3 downto 0);
    o_out: out    std_logic_vector(0 to 6)

  );
end entity;
```


Escreva o decodificador em VHDL

```
architecture ckt of hex_to_7seg is

begin
    process(i_digit)
    begin
        case i_digit is
            when "0000" => o_out <= not "1111110";
            when "0001" => o_out <= not "0110000";
            when "0010" => o_out <= not "1101101";
            when "0011" => o_out <= not "1111001";
            when "0100" => o_out <= not "0110011";
            when "0101" => o_out <= not "1011011";
            when "0110" => o_out <= not "1011111";
            when "0111" => o_out <= not "1110000";
            when "1000" => o_out <= not "1111111";
            when "1001" => o_out <= not "1111011";
            when "1010" => o_out <= not "1110111";
            when "1011" => o_out <= not "0011110";
            when "1100" => o_out <= not "1001110";
            when "1101" => o_out <= not "0111101";
            when "1110" => o_out <= not "1001111";
            when "1111" => o_out <= not "1000111";

            when others => o_out <= "0000000";
        end case;
    end process;

end ckt;
```

Create a testbench

Neste caso, como o circuito é muito simples , vamos direto para a implementação

Implementação na placa

- Crie um novo arquivo VHDL hex_to_7seg_de10

```
library ieee;
use ieee.std_logic_1164.all;

entity hex_to_7seg_de10 is

    port(

        SW:          in          std_logic_vector(3 downto 0);
        HEX0: out       std_logic_vector(0 to 6)

    );
end entity;
```

Implementação na placa

- Crie um novo arquivo VHDL hex_to_7seg_de10

```
architecture hardware of hex_to_7seg_de10 is

    component hex_to_7seg
        port (
            i_digit:          in      std_logic_vector(3 downto 0);
            o_out:            out      std_logic_vector(0 to 6)
        );
    end component;

    begin

        DEC: hex_to_7seg
            port map (

                                i_digit=>SW,
                                o_out=>HEX0

            );

    end hardware;
```

Implementação na placa

- Compile

The screenshot displays the Quartus Prime Lite Edition interface during the compilation of a project named 'simple_counter'. The top window shows the 'Flow Summary' for the 'simple_counter.sdc' file, indicating a successful compilation on Sun Sep 22 10:58:15 2024. The summary includes details such as the Quartus Prime version (20.1.0), revision name (simple_counter), top-level entity name (simple_counter_de10), family (MAX 10), device (10M50DAF484C7G), and various resource usage statistics.

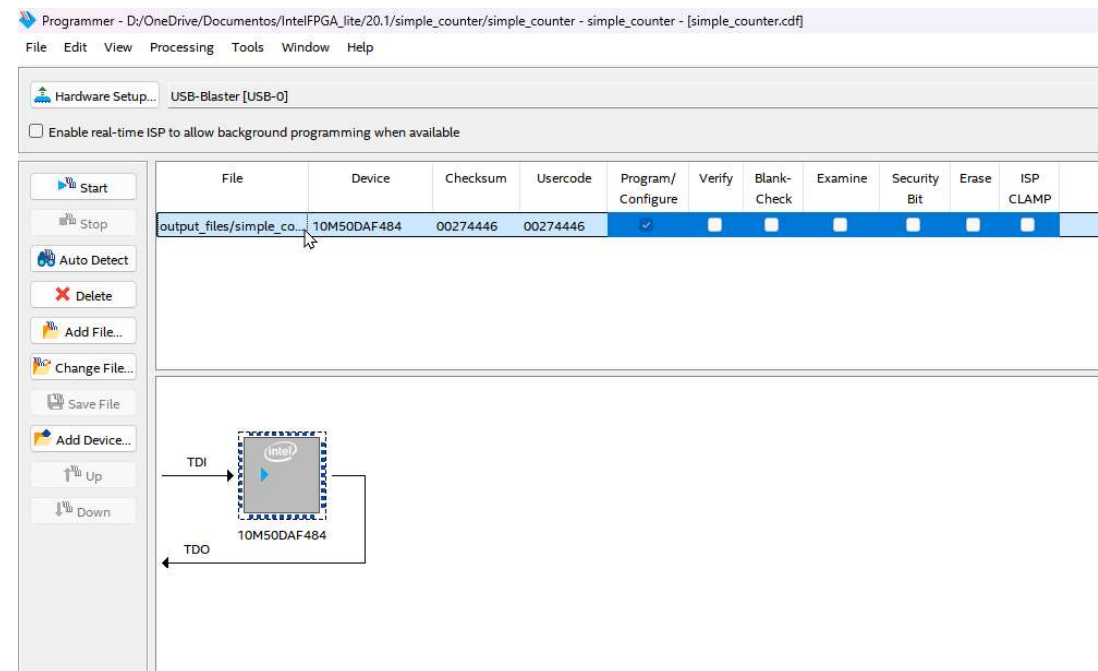
Resource	Usage	Limit	Percentage
Total logic elements	31 / 49,760	< 1 %	
Total registers	30		
Total pins	11 / 360	3 %	
Total virtual pins	0		
Total memory bits	0 / 1,677,312	0 %	
Embedded Multiplier 9-bit elements	0 / 288	0 %	
Total PLLs	0 / 4	0 %	
UFM blocks	0 / 1	0 %	
ADC blocks	0 / 2	0 %	

The bottom window shows the 'Messages' pane, which contains a list of messages generated during the compilation process. The messages include warnings about recovery paths, removal paths, and timing constraints, as well as a successful compilation status.

```
332140 No Recovery paths to report
332140 No Removal paths to report
332146 worst-case minimum pulse width slack is 9.466
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 196 warnings
Running Quartus Prime EDA Netlist writer
Command: quartus_eda --read_settings_files=off --write_settings_files=off simple_counter -c simple_counter
18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
204019 Generated file simple_counter.vho in folder "D:/OneDrive/Documents/IntelFPGA_lite/20.1/simple_counter/simulation/modelsim/" for EDA simulation tool
Quartus Prime EDA Netlist writer was successful. 0 errors, 1 warning
293000 Quartus Prime Full compilation was successful. 0 errors, 579 warnings
```

Implementação na placa

- Programe a placa
 - Conecte o DE10 ao computador
 - Tools → Programmer
 - Selectio o arquivo xxxx.sof
 - Start



Contador Simples de 4 bits

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity simple_counter is
    generic(
        n: natural := 4
    );
    port(
        i_clk: in std_logic;
        i_rstb: in std_logic;

        o_cnt : out std_logic_vector(n-1 downto 0)
    );
end entity;
```

Contador Simples de 4 bits

```
architecture behavioral of simple_counter is

    -- internal signal
    signal cnt_sig: unsigned(n-1 downto 0);

begin

    process(i_clk, i_rstb)
    begin
        if (i_rstb = '0') then
            cnt_sig <= (others => '0');
        elsif (rising_edge(i_clk) ) then
            cnt_sig <= cnt_sig + 1;

        end if;
    end process;

    o_cnt <= std_logic_vector(cnt_sig);
end behavioral;
```


Contador Simples de 4 bits – testbench

```
library ieee;
use ieee.std_logic_1164.all;

entity simple_counter_tb is
    generic ( N:natural :=4);
end entity;

architecture testbench of simple_counter_tb is
    signal CLK: std_logic;
    signal RSTB: std_logic;

    signal CNT: std_logic_vector((N-1) downto 0);

    constant PER: time := 20 ns;

    component simple_counter
        generic( n: NATURAL := 4);
        port
        (
            i_rstb    : IN STD_LOGIC;
            i_clk     : IN STD_LOGIC;
            o_cnt     : OUT STD_LOGIC_VECTOR((n-1) downto 0)

        );
    end component;
```

Contador Simples de 4 bits – testbench

```
begin

    DUT:    simple_counter
            generic map ( n=>N)

            port map (

                                i_clk    => CLK,
                                i_rstb=> RSTB,
                                o_cnt=> CNT

                                );

    --clock process

    clock: process
    begin
        CLK <='0';
        wait for PER/2;
        infinite:loop
            CLK <= not CLK; wait for PER/2;
        end loop;

    end process;
```

Contador Simples de 4 bits – testbench

```
-- Reset process
reset: process
begin
    RSTB <= '0'; wait for 2*PER;

    RSTB <= '1'; wait;
end process reset;

--run process
run:process
begin

    wait for 4*PER;
    -- run code
    wait for (2**N)*PER;

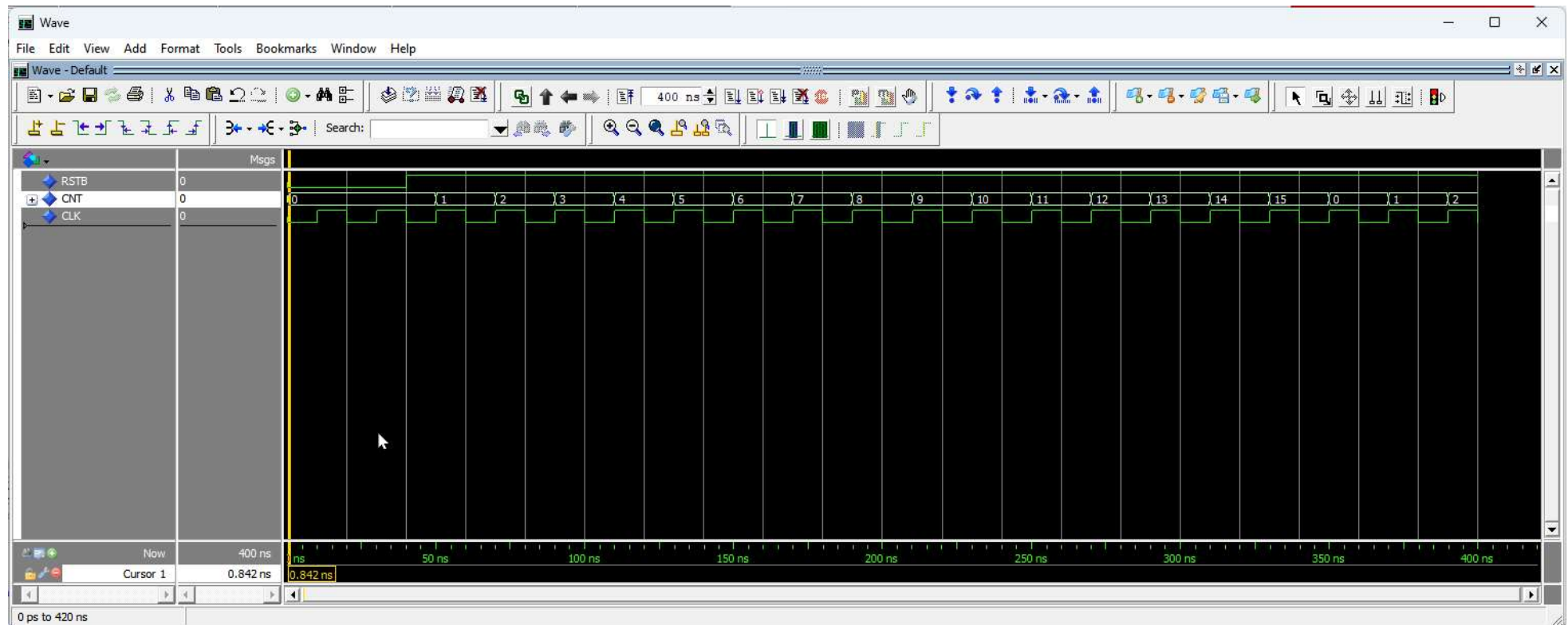
end process run;

end architecture;
```

Contador Simples de 4 bits – Run simulation

- Tools → Run simulation Tools → RTL simulation
- No modelsim
 - Compile
- Em work
 - Seleciona o arquivo de testbench
- Em objects
 - Seleciona os sinais
 - Add wave

Contador Simples de 4 bits – Run simulation



Contador Simples de 4 bits – implementação na placa

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.ALL;

entity simple_counter_de10 is
    port(
        CLOCK_50:    in    std_logic;
        SW:           in    std_logic_vector(1 downto 0);
        LEDR:         out   std_logic_vector(7 downto 0)
    );
end entity;
```

Contador Simples de 4 bits – implementação na placa

```
architecture hardware of simple_counter_de10 is

    signal CLK_SIG: std_logic; -- intermediate clock signal

    -- Component prototypes

    component clk_3Hz
        port(
            i_clk_50Mhz      : IN    STD_LOGIC;
            i_rstb            : IN    STD_LOGIC;
            o_clk_3Hz        : OUT   STD_LOGIC
        );
    end component;

    component simple_counter
        generic( n: NATURAL := 4 );
        port
        (
            i_rstb      : IN STD_LOGIC;
            i_clk        : IN STD_LOGIC;
            o_cnt        : OUT STD_LOGIC_VECTOR((n-1) downto 0)
        );
    end component;
```

Contador Simples de 4 bits – implementação na placa

```
begin
```

```
    CK:clk_3Hz
        port map
        (
            i_clk_50Mhz => CLOCK_50,
            i_rstb      => SW(0),
            o_clk_3Hz   => CLK_SIG

        );
```

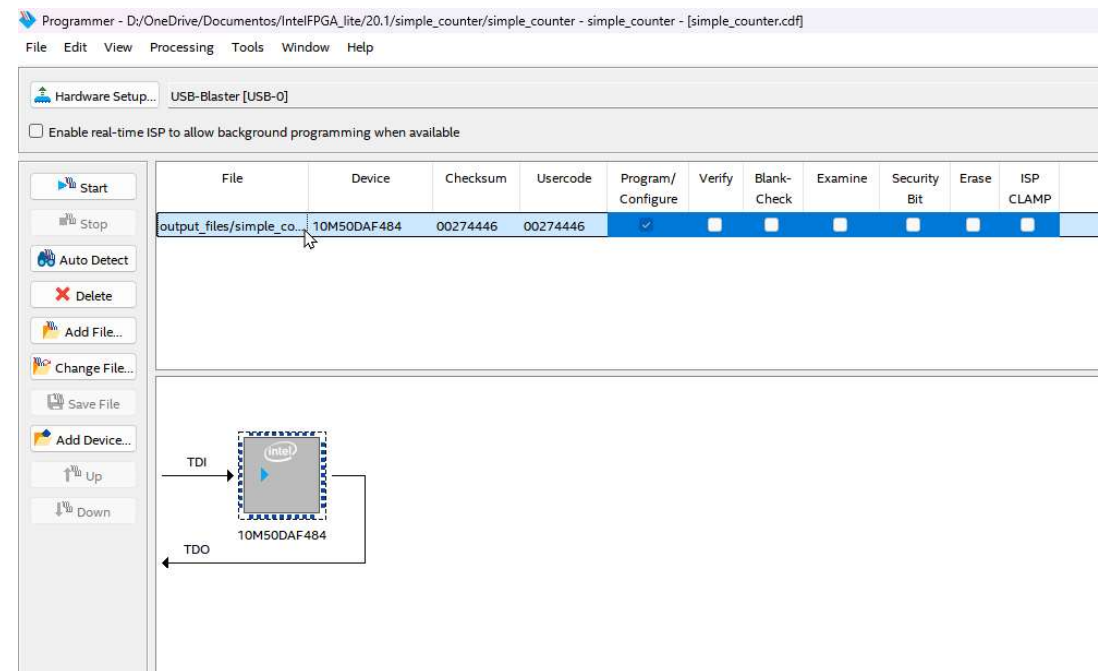
```
    DUT: simple_counter
        port map
        (
            i_clk => CLK_SIG,
            i_rstb => SW(0),
            o_cnt => LEDR( 3 downto 0)

        );
```

```
end architecture;
```


Contador Simples de 4 bits – implementação na placa

- Programe a placa
 - Conecte o DE10 ao computador
 - Tools → Programmer
 - Selectio o arquivo xxxx.sof
 - Start



Exercícios

- Faça o simple counter mostrar a contagem no display de 7 segmentos
- Implemente um contador com enable. Se o enable estiver em 0, a contagem deve ficar bloqueada (no mesmo valor).
- Implemente um contador com direção (DIR). Se $DIR = 0$ contagem crescente, se $DIR = 1$, contagem decrescente.