PRÁTICA DE ELETRÔNICA DIGITAL 1 (FGA0071)

Apresentação Placa FPGA AX301 com Quartus II

Prof. Marcelino Andrade

Semestre 2024.2

Universidade de Brasília
UnB

Sumário

Do que se trata?

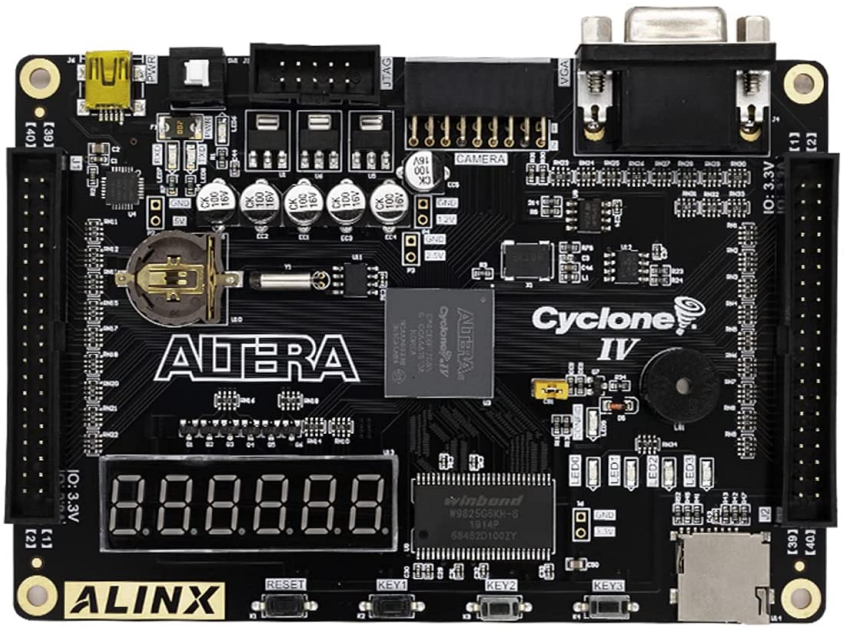


Figura 1. Placa AX301 FPGA

Figura 2. Quartus II Versão 13.1

Recursos da Placa AX301

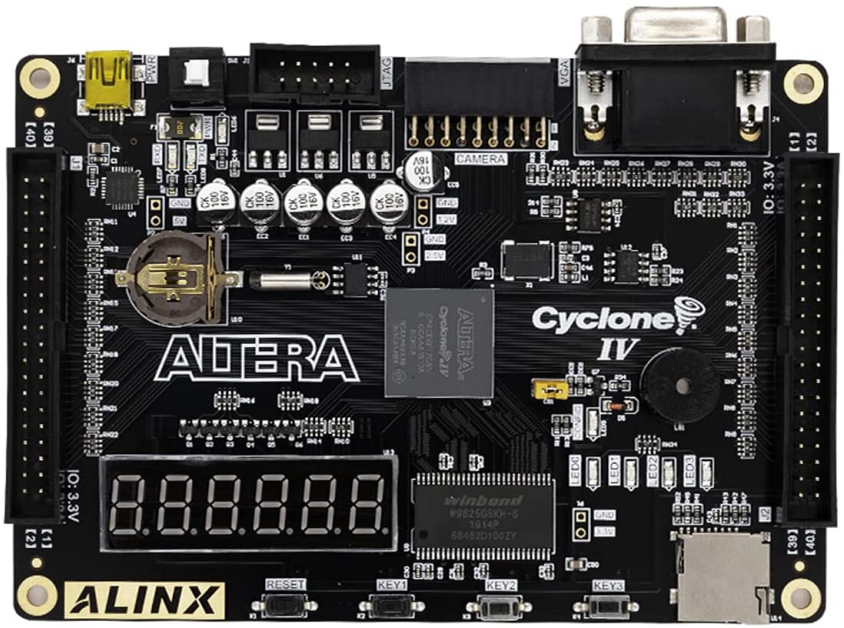


Figura 3. Placa AX301 FPGA

- Alimentação e comunicação USB.
- Memória SDRAM de 256Mbit.
- Memória SPI FLASH de 16Mbit.
- Interface para câmera.
- Interface VGA de 16 bits.
- Relógio de tempo real (RTC).
- EEPROM 24LC04 com interface IIC.
- 4 LEDs vermelhos.
- 4 botões.
- Cristal ativo de 50 MHz.
- Duas portas de expansão.
- JTAG para depuração/programação.
- Slot para cartão Micro SD.
- Display de 6 dígitos.

O Software aplicado

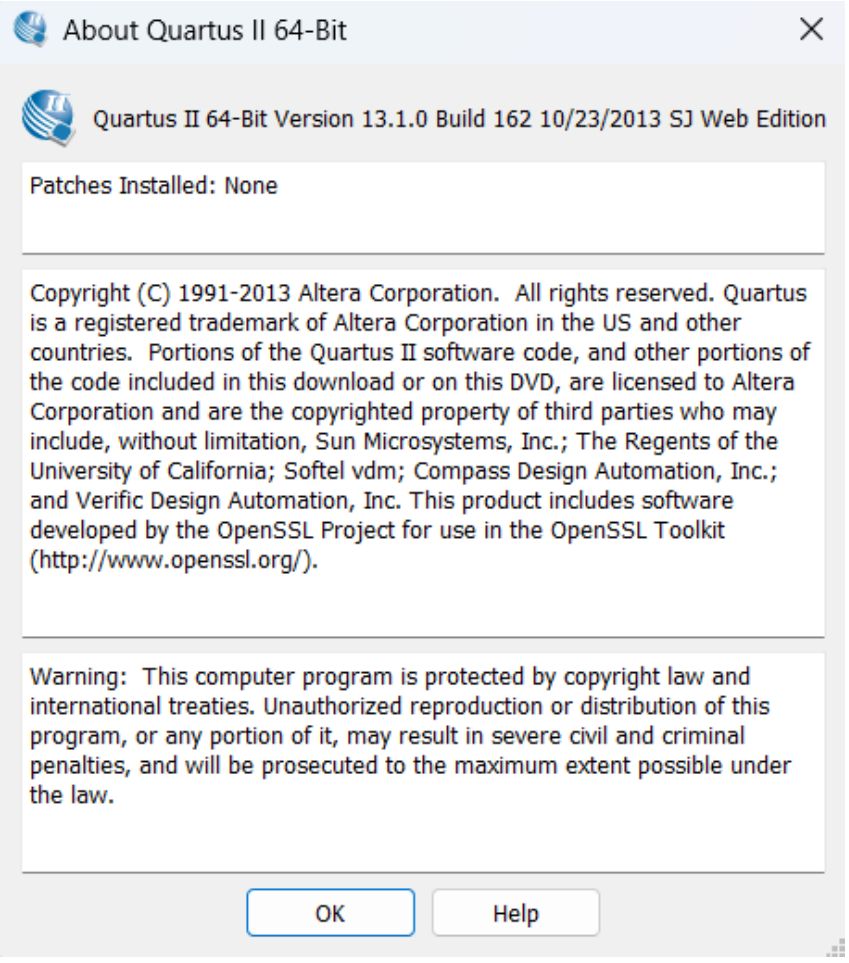


Figura 4. Detalhes da versão do Quartus II utilizada.

O software **Quartus II** é uma ferramenta essencial para a configuração e o desenvolvimento de projetos FPGA, oferecendo suporte desde a criação de projetos até a sua compilação e simulação. A versão utilizada neste guia é a **Quartus II 64-Bit Version 13.1.0 Build 162**, como mostrado na **Figura 4**. A seguir estão os passos organizados para a instalação e configuração do software.

Download e Configuração

1. **Download do Software**
   - Acesse o site oficial da **Intel**: Intel Quartus II Web Edition.
   - Faça o download do arquivo compactado no formato `.tar`.
   - Após o download, descompacte o arquivo em um diretório.
   - Execute o instalador clicando duas vezes no arquivo `setup`.

2. **Configuração da Instalação**
   - Abra o instalador e clique em **Next** para iniciar a instalação.
   - Leia e aceite - **I accept the agreement** e **Next**.
   - Escolha o diretório onde o software será instalado e **Next**.
   - Garanta opções padrão, como **Web Edition** e **Next**.

3. **Conclusão da Instalação**
   - Aguarde até que o processo de instalação seja concluído.
   - Na tela final, clique em **Finish** para encerrar o assistente.

O Projeto do Detector de Paridade

Entradas	LED0	LED1
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

- Os botões **KEY1**, **KEY2** e **KEY3** são utilizados como entradas.
- A paridade é calculada usando a operação XOR, definida como:

$$\text{Paridade} = \text{KEY1} \oplus \text{KEY2} \oplus \text{KEY3}$$

- **LED0** acende se o número de bits **"1"** for par e **LED1** sendo ímpar.

Tela Inicial do Quartus II

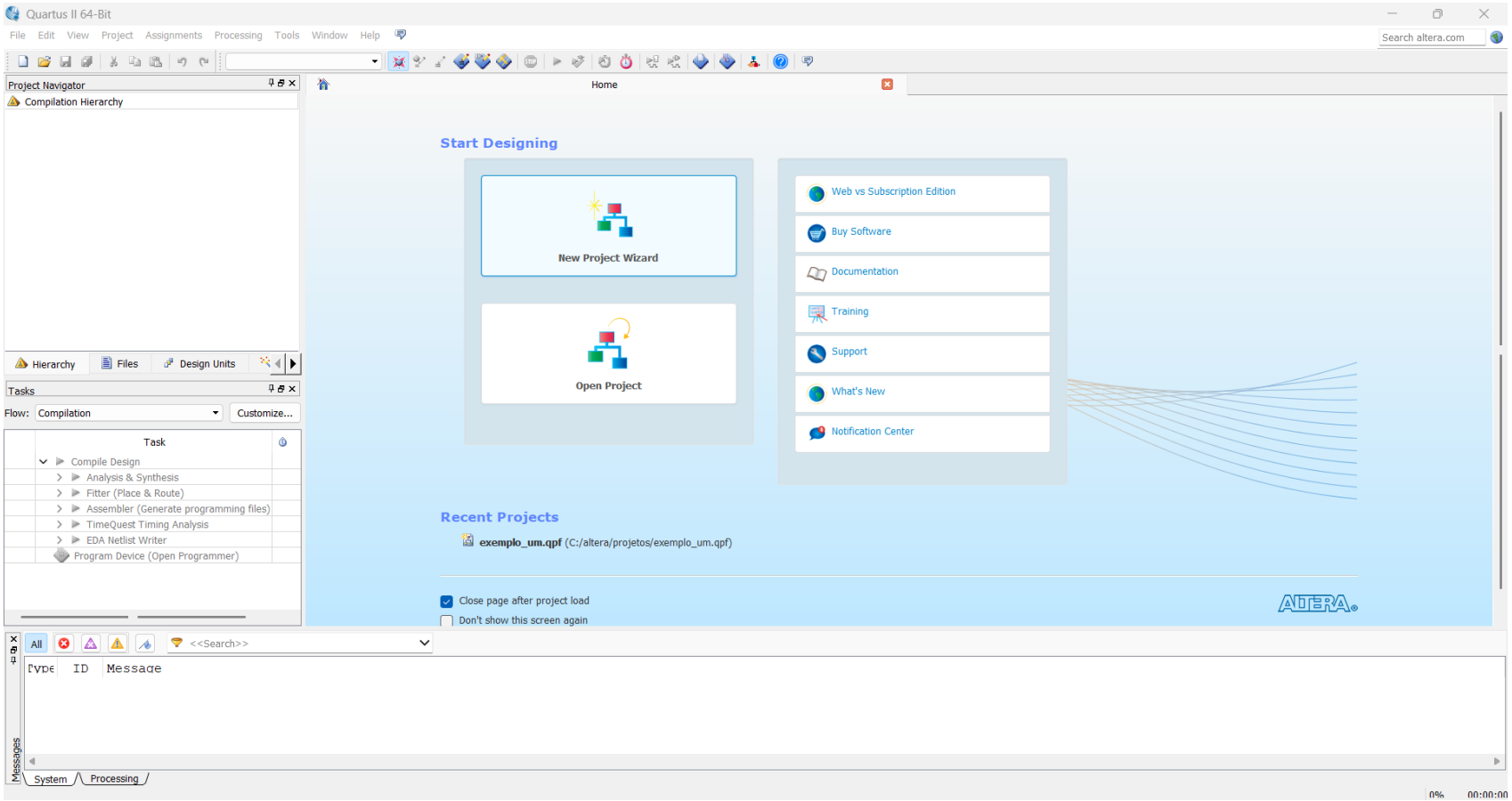


Figura 5. Tela inicial do Quartus II e opção de criação de um novo projeto

Introdução, Configuração do Diretório e Nome

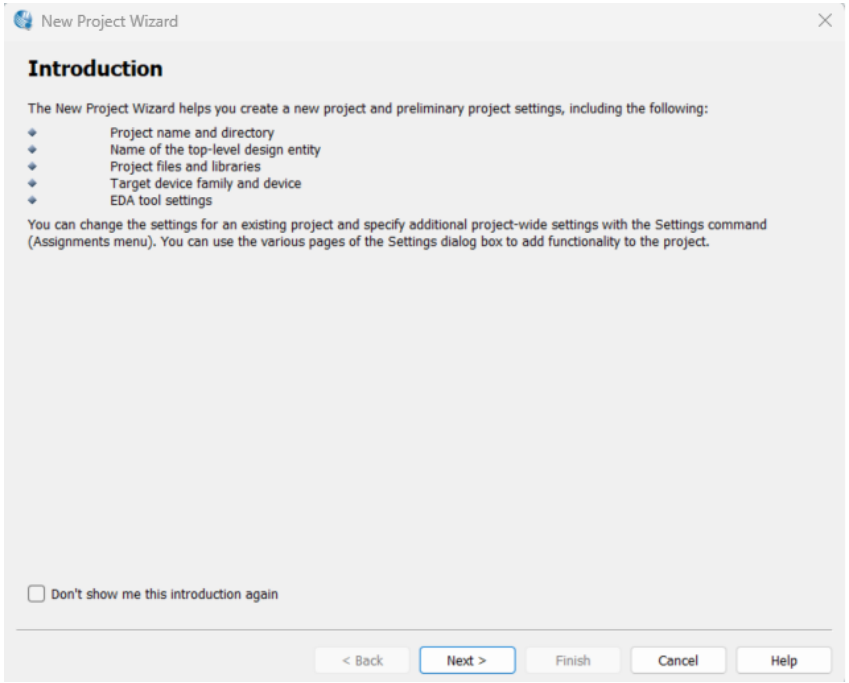


Figura 6. Introdução

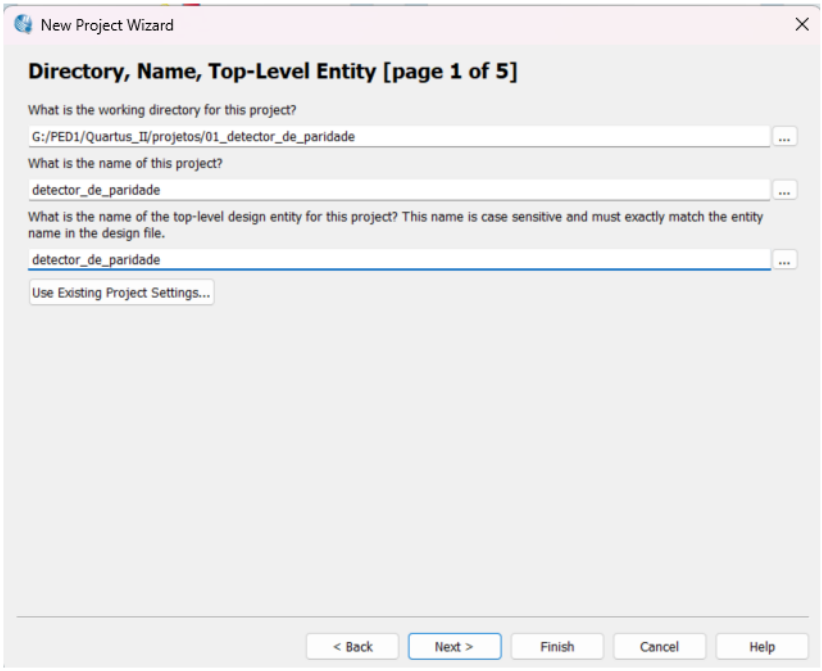


Figura 7. Diretório

Adição de Arquivos e Escolha do FGA

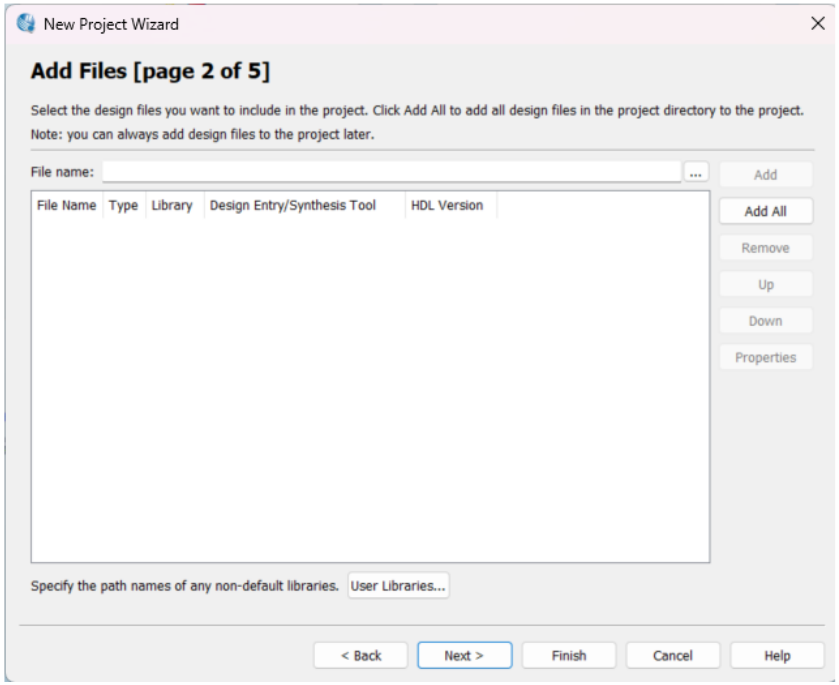


Figura 8. Adição de Arquivos

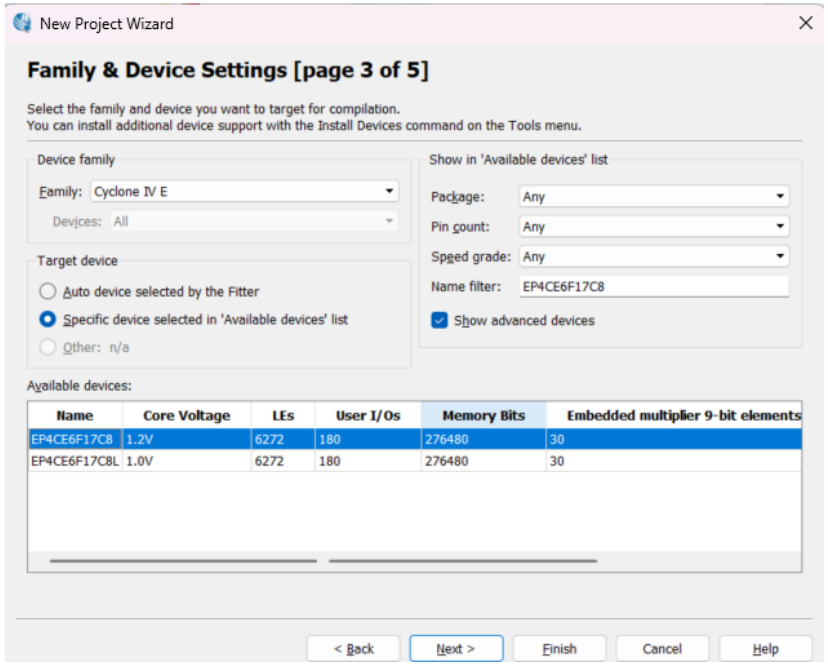


Figura 9. Escolha FPGA

Ferramentas EDA e Resumo Final

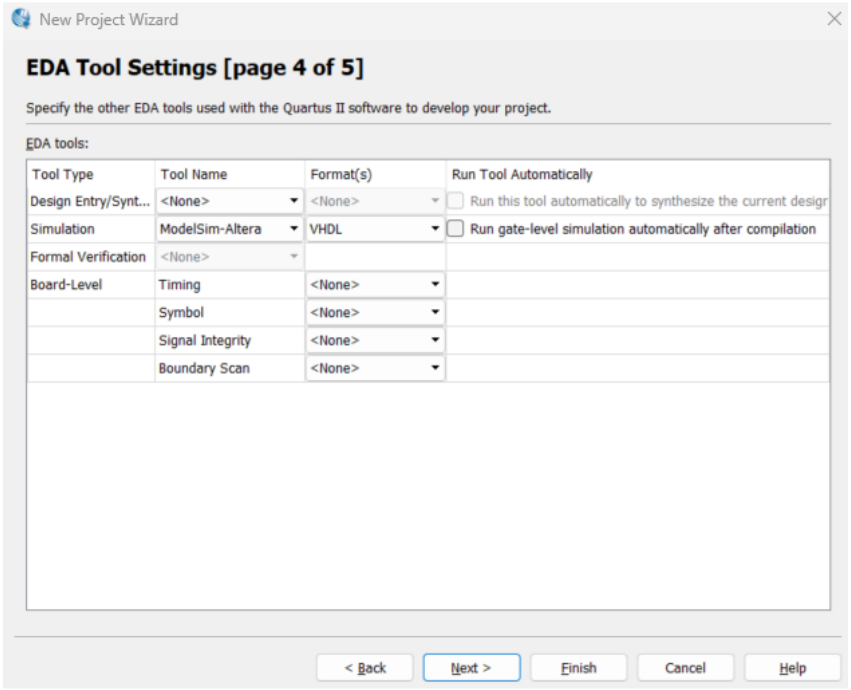


Figura 10. Ferramentas EDA

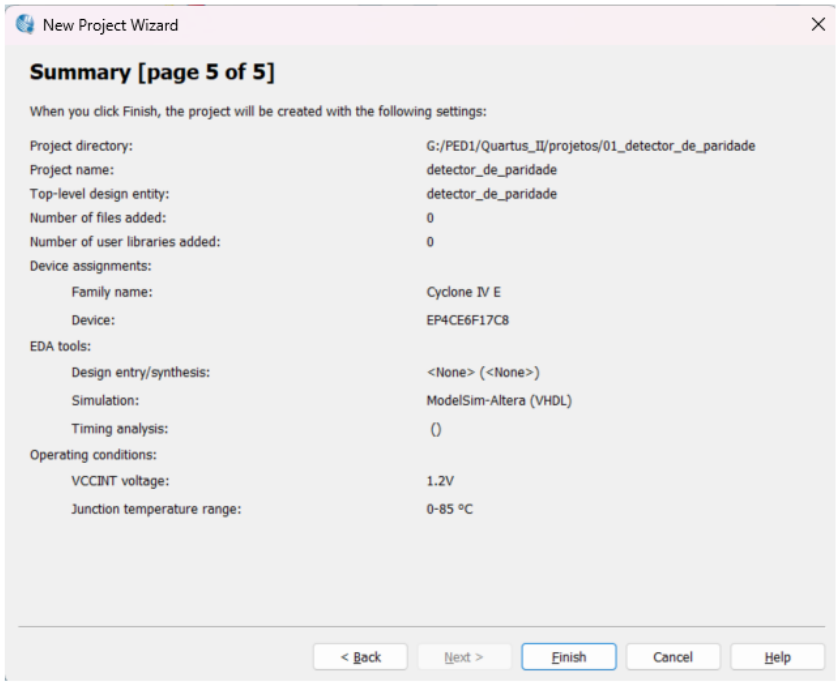


Figura 11. Resumo

Criação do Arquivo VHDL

1. No menu superior do Quartus II, clique em **File** → **New**.
2. Selecione **VHDL File** e clique em **OK**.
3. Adicione o código VHDL no editor e salve com a extensão `.vhd`.

O código apresentado ao lado representa o código VHDL do **Detector de Paridade**.

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY detector_de_paridade IS
    PORT (
        key1 : IN STD_LOGIC;   -- Entrada do botão KEY1
        key2 : IN STD_LOGIC;   -- Entrada do botão KEY2
        key3 : IN STD_LOGIC;   -- Entrada do botão KEY3
        led0 : OUT STD_LOGIC;  -- Saída do LED0 (Paridade Par)
        led1 : OUT STD_LOGIC   -- Saída do LED1 (Paridade Ímpar)
    );
END detector_de_paridade;

ARCHITECTURE behavior OF detector_de_paridade IS
    SIGNAL paridade : STD_LOGIC;  -- Sinal intermediário
BEGIN
    PROCESS (key1, key2, key3)
    BEGIN
        -- Calcular paridade usando XOR
        paridade <= key1 XOR key2 XOR key3;
        -- Atribuir resultados aos LEDs
        led0 <= NOT paridade;  -- LED0  para paridade par
        led1 <= paridade;      -- LED1  para paridade ímpar
    END PROCESS;
END behavior;
```

Figura 12. Código Detector de Paridade

Compilação do Código VHDL

- Clique em **Start Compilation**.
- Em caso de sucesso, as mensagens na janela **Task** em verdes.

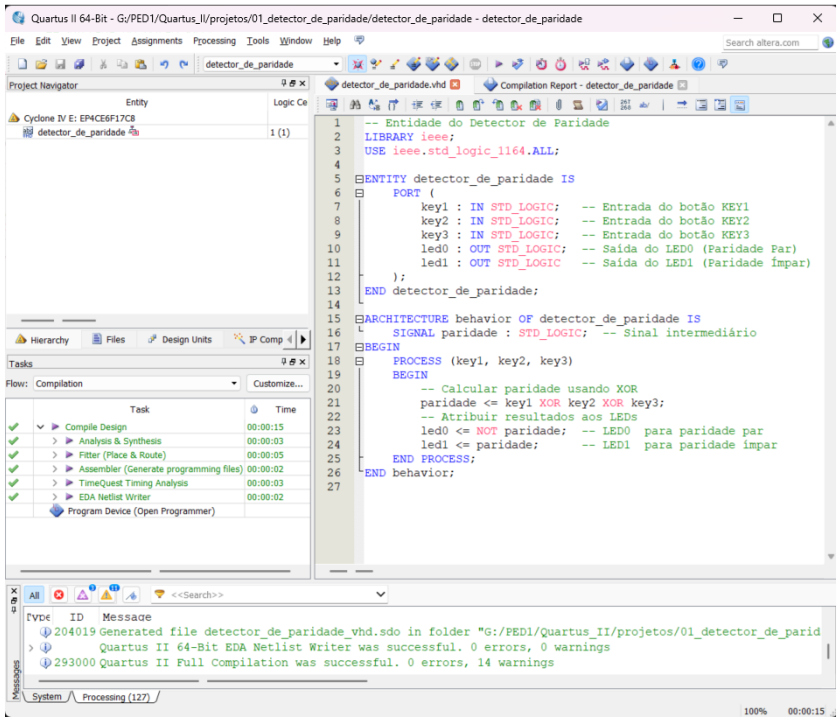


Figura 13. Conclusão bem-sucedida de todas as etapas de compilação no Quartus II

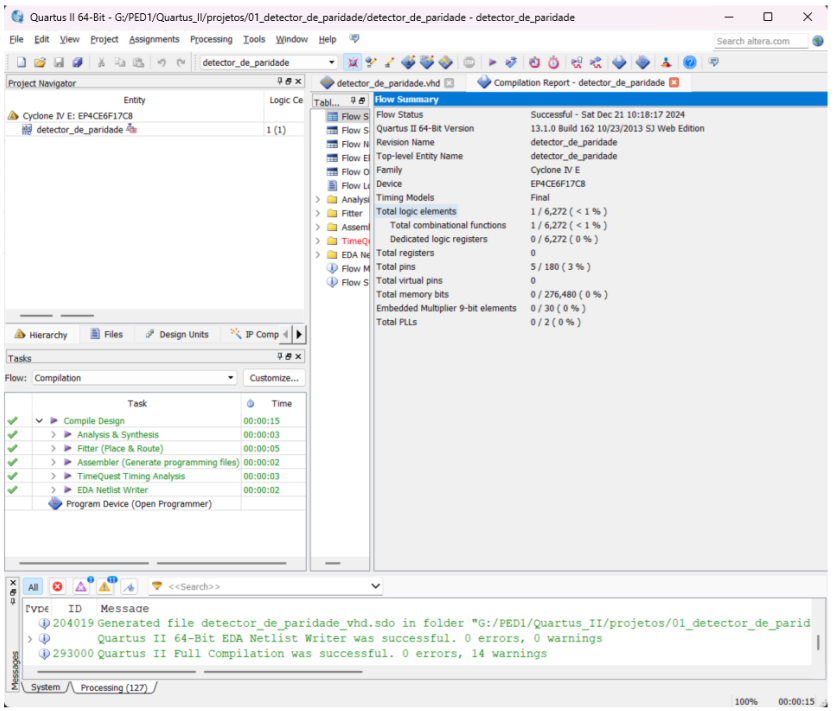


Figura 14. Flow Summary apresenta os detalhes técnicos da compilação

Visualização no RTL Viewer

- Acesse **Tools → Netlist Viewers → RTL Viewer**.
- Aguarde o carregamento da visualização lógica do circuito.

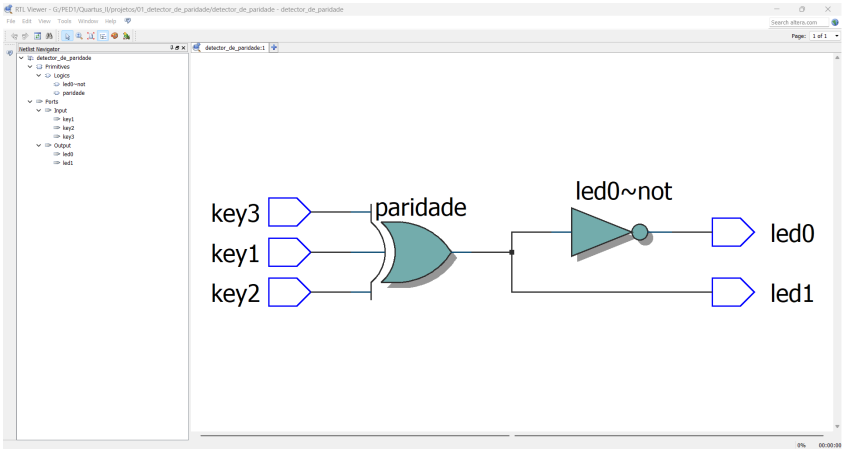


Figura 15. Visão lógica do circuito no RTL Viewer, destacando os componentes principais: entradas (`key1`, `key2`, `key3`), a operação lógica XOR (sinal `paridade`) e as saídas (`led0`, `led1`).

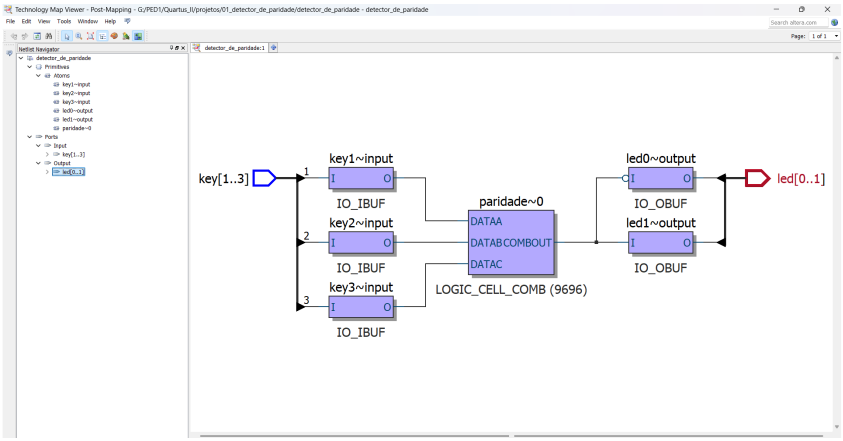
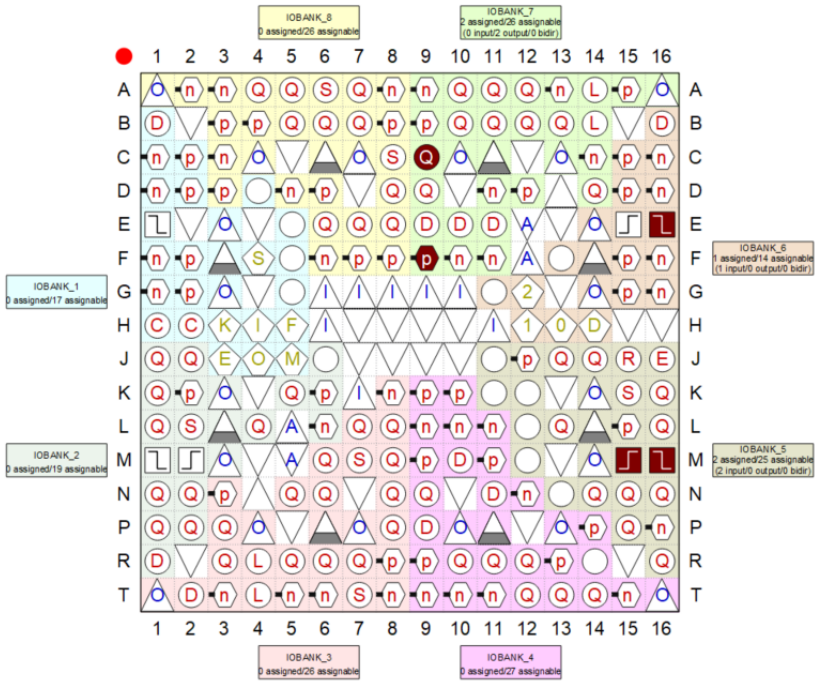


Figura 16. Visão pós-mapeamento no Technology Map Viewer, evidenciando as conexões físicas dos sinais de entrada e saída no FPGA, incluindo os buffers de entrada (`IO_IBUF`) e saída (`IO_OBUF`).

Mapeamento de Pinos

1. Acesse **Assignments** → **Pin Planner**.
2. Consulte as tabelas 1 e 2, desse guia, para selecionar os pinos.
3. Insira os pinos na coluna **Location**.
4. Compile novamente o projeto clicando em **Start Compilation**.

O *Pin Planner* permite associar os sinais lógicos do design às portas físicas do dispositivo.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
key1	Input	PIN_M15	5	B5_N0	PIN_F3	2.5 V (default)		8mA (default)		
key2	Input	PIN_M16	5	B5_N0	PIN_G5	2.5 V (default)		8mA (default)		
key3	Input	PIN_E16	6	B6_N0	PIN_J1	2.5 V (default)		8mA (default)		
led0	Output	PIN_F9	7	B7_N0	PIN_B1	2.5 V (default)		8mA (default)	2 (default)	
led1	Output	PIN_C9	7	B7_N0	PIN_F2	2.5 V (default)		8mA (default)	2 (default)	
<<new node>>										

Figura 17. Visualização do *Pin Planner* no Quartus II, mostrando a atribuição de pinos do FPGA Cyclone IV E (EP4CE6F17C8)

Embarque do Código na FPGA

1. No menu, clique em **Tools** → **Programmer**.
2. Adicione o arquivo `.sof` na pasta **output_files**.
3. Certifique-se de que o *USB-Blaster* foi reconhecido.
4. Clique em **Start** e aguarde o progresso até 100%.

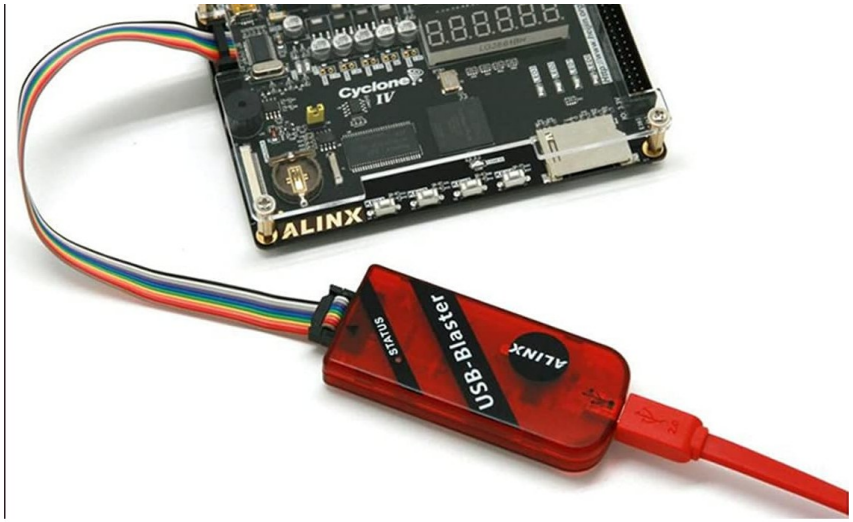


Figura 18. Conexão e programação da placa FPGA AX301 utilizando o *USB-Blaster*.

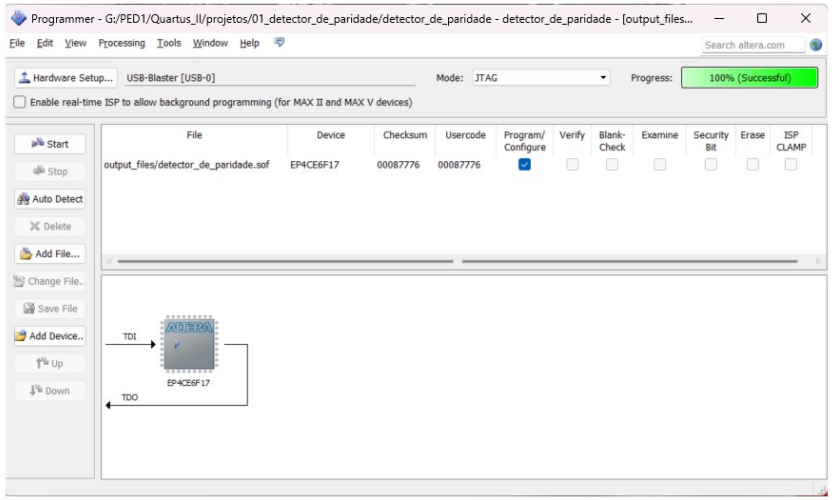


Figura 19. *Quartus II Programmer*.

Configuração do ModelSim

1. Abra o Quartus II e acesse o menu **Tools** → **Options**.
2. Selecione a aba **EDA Tool Options**.
3. Configure o caminho do executável do ModelSim na seção **ModelSim-Altera**.
4. Clique em **OK** para salvar as configurações.

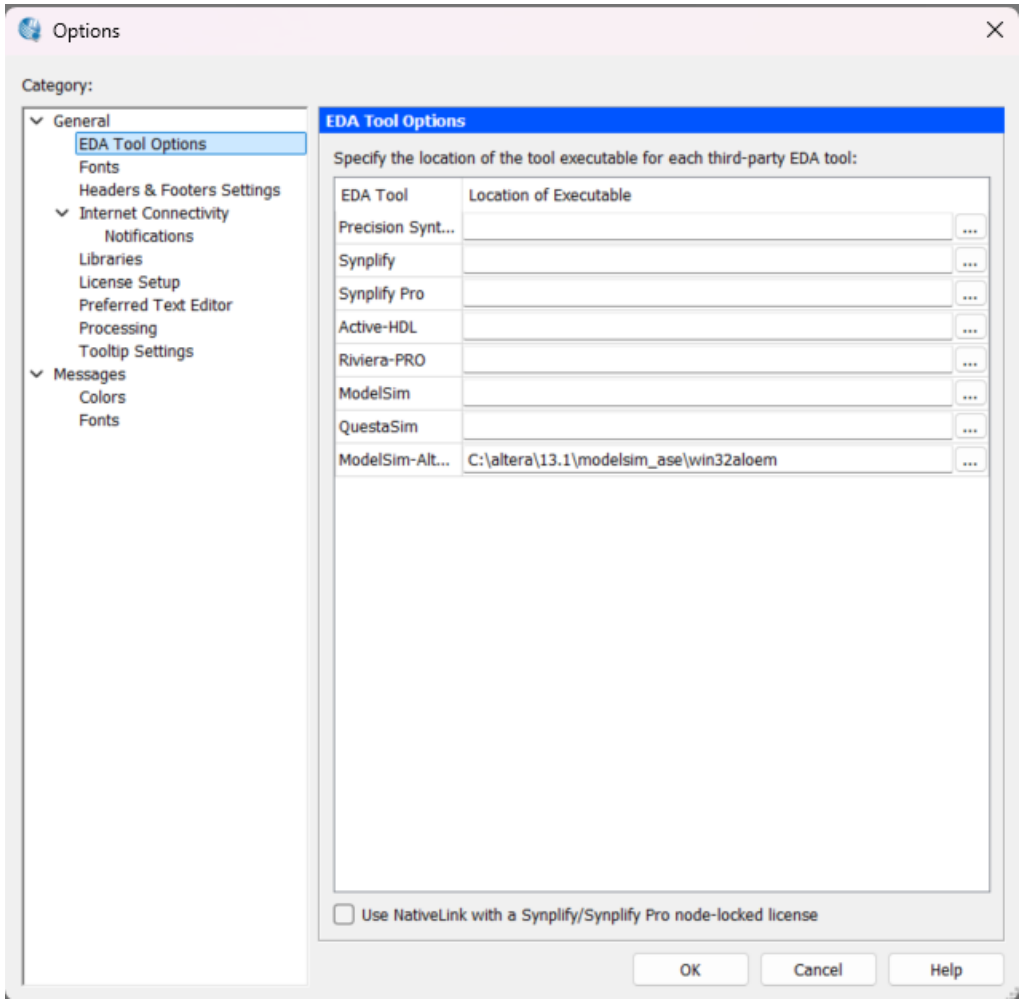


Figura 20. Caminho para o executável do ModelSim.

Criação do Testbench em VHDL

- Um **Testbench** é um código responsável por gerar estímulos para testar o projeto.
- Crie um novo arquivo VHDL no Quartus II seguindo os passos:

1. No menu superior, clique em **File** → **New**.
2. Selecione **VHDL File** e clique em **OK**.
3. Adicione o código abaixo e salve como **tb_detector_de_paridade.vhd**.

Código do Testbench em VHDL

```vhdl
-- Testbench para Detector de Paridade
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_detector_de_paridade IS
END tb_detector_de_paridade;

ARCHITECTURE behavior OF tb_detector_de_paridade IS

    -- Componentes a serem testados
    COMPONENT detector_de_paridade
        PORT (
            key1 : IN STD_LOGIC;
            key2 : IN STD_LOGIC;
            key3 : IN STD_LOGIC;
            led0 : OUT STD_LOGIC;
            led1 : OUT STD_LOGIC
        );
    END COMPONENT;

    -- Sinais de estímulo
    SIGNAL key1 : STD_LOGIC := '0';
    SIGNAL key2 : STD_LOGIC := '0';
    SIGNAL key3 : STD_LOGIC := '0';
    SIGNAL led0 : STD_LOGIC;
    SIGNAL led1 : STD_LOGIC;

BEGIN

    -- Instância do componente
    uut: detector_de_paridade PORT MAP (
        key1 => key1,
        key2 => key2,
        key3 => key3,
        led0 => led0,
        led1 => led1
    );

    -- Processo de estímulo
```

Figura 21. Código Testbench Parte 1/2

```vhdl
    stim_proc: PROCESS
    BEGIN
        -- Teste para entrada 000
        key1 <= '0'; key2 <= '0'; key3 <= '0';
        WAIT FOR 10 ns;

        -- Teste para entrada 001
        key1 <= '0'; key2 <= '0'; key3 <= '1';
        WAIT FOR 10 ns;

        -- Teste para entrada 010
        key1 <= '0'; key2 <= '1'; key3 <= '0';
        WAIT FOR 10 ns;

        -- Teste para entrada 011
        key1 <= '0'; key2 <= '1'; key3 <= '1';
        WAIT FOR 10 ns;

        -- Teste para entrada 100
        key1 <= '1'; key2 <= '0'; key3 <= '0';
        WAIT FOR 10 ns;

        -- Teste para entrada 101
        key1 <= '1'; key2 <= '0'; key3 <= '1';
        WAIT FOR 10 ns;

        -- Teste para entrada 110
        key1 <= '1'; key2 <= '1'; key3 <= '0';
        WAIT FOR 10 ns;

        -- Teste para entrada 111
        key1 <= '1'; key2 <= '1'; key3 <= '1';
        WAIT FOR 10 ns;

        -- Fim da simulação
        WAIT;
    END PROCESS;

END behavior;
```

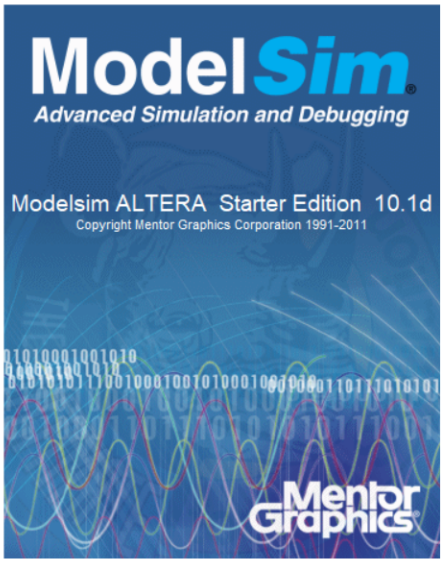
Figura 22. Código Testbench Parte 2/2

Execução da Simulação no ModelSim

1. No Quartus II, clique em **Tools** → **Run Simulation Tool** → **RTL Simulation**. O ModelSim será aberto automaticamente.
2. No ModelSim, compile o Testbench e o código do projeto:
   - Digite **vcom -93 detector_de_paridade.vhd** no console para compilar o código VHDL do design principal.
   - Digite **vcom -93 tb_detector_de_paridade.vhd** no console para compilar o Testbench associado.
3. Inicie a simulação carregando o Testbench com o comando:
   - **vsim work.tb_detector_de_paridade**
4. Adicione as formas de onda desejadas à interface gráfica:
   - **add wave ***
5. Execute a simulação para um período de 100 ns com:
   - **run 100 ns**

   As formas de onda dos sinais serão geradas na janela de simulação.

Tela do Ambiente de Simulação



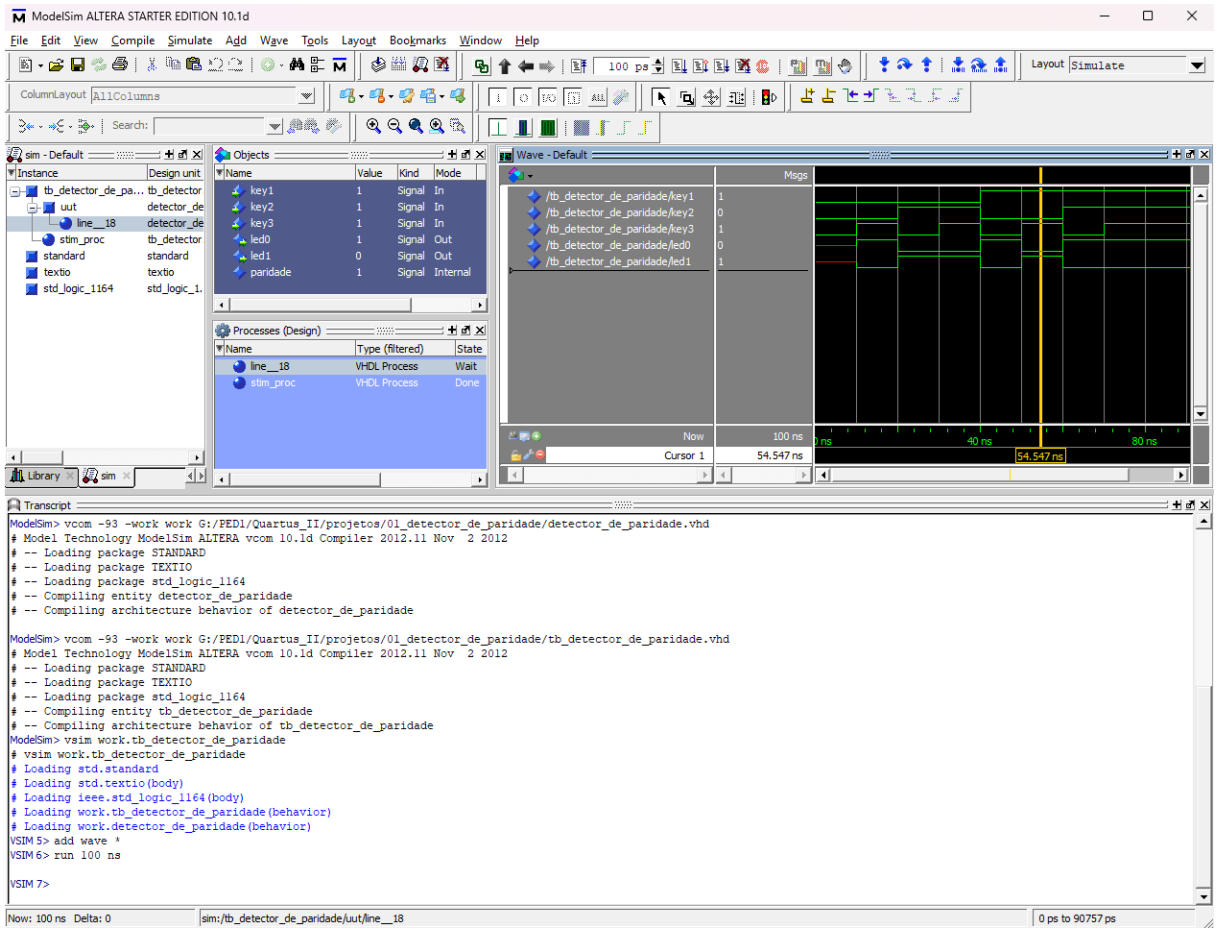


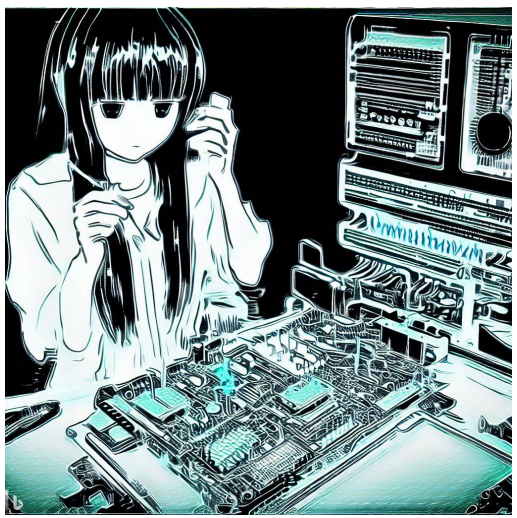
Figura 23. A imagem à esquerda mostra a tela de abertura do ModelSim ALTERA Starter Edition 10.1d, indicando a versão do software utilizada. A imagem à direita apresenta o ambiente de simulação com a exibição dos comandos utilizados e as formas de onda dos sinais de entrada e saída.

Onde chegamos e podemos chegar?

- Os passos essenciais para o uso da placa FPGA AX301 com Quartus II e ModelSim foram apresentados:
  - Instalação, configuração e desenvolvimento de exemplo prático.
  - Criação, simulação e embarque de designs em VHDL.
- Próximos passos naturais:
  - Integração com sensores externos e displays.
  - Uso de memória interna para armazenamento de dados.
  - Uso de interfaces externas (UART, VGA e SDCard).
- Domínio de ferramentas (Quartus II e ModelSim) permite:
  - Criação de sistemas embarcados.
  - Processamento de sinais digitais.
- Este manual é um ponto de partida para capacitar estudantes e profissionais em projetos com FPGAs.

Fim!

Obrigado pela Atenção!



Prof. Marcelino Andrade
andrade@unb.br