

# Guia Técnico

## Instalação e Uso da Placa FPGA

## AX301 com Quartus II

Version 1.0

Prof. Marcelino Monteiro de Andrade

Campus UnB Gama  
Faculdade de Ciências e Tecnologias em Engenharias - FCTE  
4 de fevereiro de 2025

# Sumário

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Apresentação da Placa AX301</b>	<b>2</b>
2.1	Alimentação da Placa . . . . .	3
2.2	FPGA Cyclone IV . . . . .	3
2.3	Memória . . . . .	5
2.4	Interfaces . . . . .	6
2.5	Periféricos . . . . .	7
2.6	Portas de Expansão . . . . .	11
<b>3</b>	<b>Instalação do Software Quartus II</b>	<b>12</b>
3.1	Download do Software . . . . .	12
3.2	Configuração da Instalação . . . . .	13
3.3	Conclusão da Instalação . . . . .	14
<b>4</b>	<b>Desenvolvimento de Projeto no Quartus II</b>	<b>14</b>
4.1	Projeto Detector de Paridade . . . . .	15
4.2	Criação de um Novo Projeto . . . . .	16
4.3	Criação do Arquivo VHDL . . . . .	16
4.4	Compilação do Código VHDL . . . . .	18
4.5	Visualização no RTL Viewer . . . . .	18
4.6	Mapeamento de Pinos . . . . .	20
4.7	Embarque do Código na FPGA . . . . .	20
<b>5</b>	<b>Simulação com ModelSim e GHDL</b>	<b>21</b>
5.1	Configuração do ModelSim . . . . .	22
5.2	Criação do Testbench em VHDL . . . . .	23
5.3	Execução da Simulação no ModelSim . . . . .	26
5.4	Simulação com GHDL e GTKWAVE . . . . .	27
5.5	Análise dos Resultados . . . . .	30
<b>6</b>	<b>Considerações Finais</b>	<b>30</b>

# 1 Introdução

Este guia técnico, desenvolvido no Curso de Engenharia Eletrônica do Campus UnB-Gama, apresenta a **placa FPGA AX301**, a instalação e configuração do software **Quartus II**, além de um exemplo prático de implementação com **VHDL**. O documento também aborda a simulação do projeto utilizando o software **ModelSim** e o **GHD/GTKWave**, fornecendo uma base completa para o desenvolvimento e validação de projetos em FPGA a nível educacional. Além disso, as ferramentas **Quartus II**, **ModelSim** e **GHD** são amplamente documentadas na literatura técnica, como destacado em [1], [2], [3] e [4].

## 2 Apresentação da Placa AX301

A placa de desenvolvimento AX301 possui uma série de componentes principais e recursos integrados que a tornam uma plataforma ideal para aplicações educacionais e projetos introdutórios com FPGAs, além de permitir escalabilidade para soluções mais complexas e avançadas. No centro do sistema está o **FPGA Cyclone IV EP4CE6F17C8**, um modelo robusto da família Cyclone IV da **Altera** (Intel), encapsulado em formato **BGA com 256 pinos**, oferecendo alta densidade de conexões e eficiência no processamento. Essa combinação permite que estudantes desenvolvam desde projetos simples até aplicações de maior complexidade.

Para suporte ao armazenamento e à execução de dados, a placa inclui **memória SDRAM** dinâmica, ideal para aplicações com grandes volumes de dados, e **SPI Flash**, utilizada para armazenar configurações do FPGA e dados não voláteis. Além disso, há uma **EEPROM** dedicada para o armazenamento de parâmetros e configurações do usuário, facilitando o desenvolvimento de projetos customizados. A placa AX301 é especialmente adequada para experimentação e aprendizado, integrando recursos como **RTC (Relógio de Tempo Real)** para controle temporal preciso e **LEDs e chaves físicas**, que permitem visualização e interação com os estados do sistema. Esses recursos a tornam excelente para o ensino de conceitos de hardware digital e desenvolvimento de projetos práticos. Além disso, suas diversas **interfaces de expansão e conectividade**, como headers de I/O configuráveis e portas de comunicação, possibilitam a integração com periféricos externos, sensores, displays e outros dispositivos, facilitando a transição para aplicações mais sofisticadas em ambientes educacionais e profissionais.

## 2.1 Alimentação da Placa

A placa de desenvolvimento AX301 é alimentada via USB. Utilize um cabo **Mini USB** para conectar a placa ao computador e pressione o interruptor de alimentação para ligar a placa de desenvolvimento. O diagrama de projeto do circuito de alimentação é mostrado na **Figura 1**.

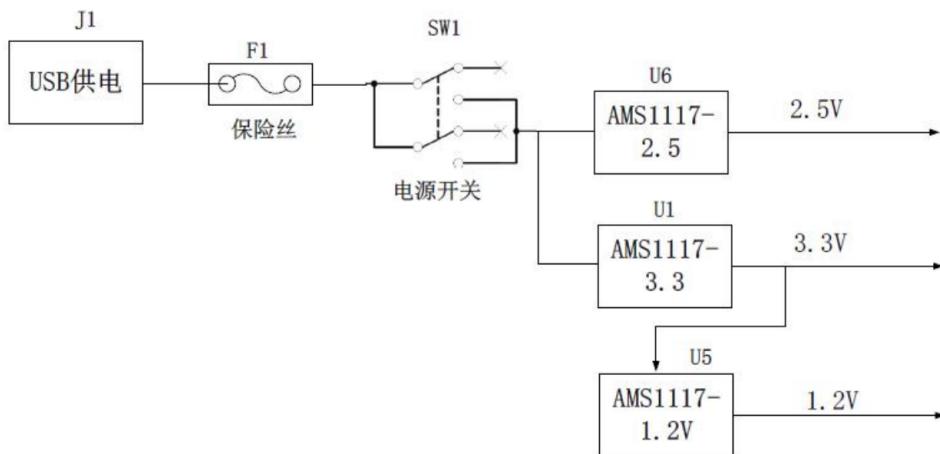


Figura 1: Diagrama do Circuito de Alimentação da Placa AX301. O esquema apresenta alimentação via USB com conversão de tensões principais: **1.2V** (Core da FPGA), **2.5V** (PLL e circuitos analógicos) e **3.3V** (bancos de I/O e periféricos) [4].

A placa gera três tensões principais: **+3,3V**, **+2,5V** e **+1,2V**, através de três chips reguladores de tensão **LDO**, que fornecem a tensão necessária para os bancos de I/O e o núcleo da FPGA. No design da **PCB**, é utilizado um layout de **4 camadas**, com camadas dedicadas para alimentação e **GND**, garantindo excelente estabilidade no fornecimento de energia para toda a placa. Pontos de teste foram reservados na PCB para que o usuário possa verificar as tensões geradas diretamente na placa.

## 2.2 FPGA Cyclone IV

A placa de desenvolvimento AX301 utiliza o modelo de FPGA **EP4CE6F17C8**, da família **Cyclone IV** da **Altera** (Intel). Este modelo é fornecido no encapsulamento **BGA (Ball Grid Array)** com **256 pinos**, oferecendo maior densidade e conectividade. Diferentemente de encapsulamentos não-BGA (como os de 144 ou 208 pinos), em que os pinos são numerados sequencialmente, no BGA

os pinos são identificados por uma combinação de **letras + números** (por exemplo, **E3, G3**). Essa nomenclatura facilita a identificação dos pinos no **layout da PCB** e no **diagrama esquemático**. A **Figura 2** ilustra o chip **Cyclone IV EP4CE6F17C8** e destaca seu encapsulamento BGA-256.



Figura 2: Chip FPGA Cyclone IV EP4CE6F17C8. Encapsulamento BGA-256 com pinos identificados por letras e números [4].

O modelo **EP4CE6F17C8** possui características principais que o tornam altamente eficiente e versátil. Ele utiliza um **encapsulamento BGA-256**, que proporciona alta densidade de conexões e maior eficiência no uso do espaço físico. Seus **bancos de I/O configuráveis** permitem suporte a diversas interfaces de entrada e saída, facilitando a integração com outros dispositivos. Além disso, conta com **suporte a PLL (Phase-Locked Loop)**, permitindo o gerenciamento eficiente de clocks para sincronização dos sistemas. Por fim, o modelo oferece **desempenho otimizado**, sendo ideal para aplicações de baixo custo com alta eficiência energética.

A utilização desse FPGA permite explorar aplicações avançadas, como controle de periféricos, interfaces de comunicação digital e processamento de sinais. O encapsulamento BGA e a disposição organizada dos pinos garantem uma integração robusta entre o FPGA e os demais componentes da placa, facilitando o desenvolvimento de projetos complexos.

## 2.3 Memória

A placa de desenvolvimento FPGA AX301 é equipada com um chip **SPI FLASH** de **16 Mbit**, modelo **W25P16**, que utiliza o padrão de tensão **CMOS 3.3V**. Devido às suas características **não voláteis**, o SPI FLASH pode ser utilizado como **imagem de boot** do sistema FPGA, permitindo o armazenamento de arquivos essenciais, como os arquivos de configuração **JIC**, código de aplicações de software e outros arquivos de dados definidos pelo usuário. As especificações do SPI FLASH estão detalhadas em [4] e o seu hardware é apresentado na **Figura 3**.

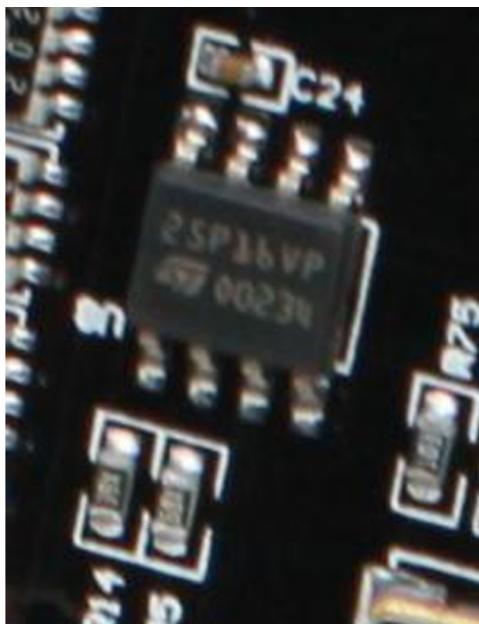


Figura 3: Chip SPI FLASH na Placa FPGA AX301 [4].

Além disso, a placa AX301 conta com um chip **SDRAM** integrado, modelo **HY57V2562GTR**, com capacidade de **256 Mbit** ( $16M \times 16\text{bit}$ ) e barramento de dados de **16 bits**. A SDRAM é amplamente utilizada para **bufferização de dados**, oferecendo armazenamento temporário para grandes volumes de informações. Um exemplo prático de uso é o armazenamento temporário dos dados coletados por uma câmera, que são armazenados na SDRAM e exibidos posteriormente por meio da interface **VGA**. Dessa forma, a SDRAM desempenha um papel essencial no **cache de dados** do sistema. O esquema de conexão da SDRAM está ilustrado na **Figura 4**, e mais detalhes sobre seu funcionamento podem ser encontrados em [4].



Figura 4: hip SDRAM na Placa FPGA AX301 [4].

Por fim, a placa AX301 possui uma **EEPROM** integrada, modelo **24LC04**, com capacidade de **4 Kbit**, organizada em dois blocos de **256 bytes**. A comunicação com a EEPROM é realizada através do barramento **IIC** (Inter-Integrated Circuit), o que facilita sua operação. Essa memória é amplamente utilizada em projetos de **instrumentação** e medição, sendo ideal para armazenar parâmetros e configurações de sistema. Graças à sua operação simples e ao excelente **custo-benefício**, a EEPROM se destaca como uma solução eficiente e econômica, especialmente em produtos que exigem alta relação entre custo e desempenho.

## 2.4 Interfaces

A placa de desenvolvimento AX301 possui diversas interfaces integradas que facilitam a comunicação e o armazenamento de dados em projetos com FPGA. A interface **USB-UART** é implementada pelo chip **CP2102GM** da **Silicon Labs**. Ela utiliza um conector **Mini USB**, que, além de fornecer alimentação à placa, permite a conversão da porta USB em uma porta serial para comunicação de dados. Com o uso de um cabo USB, a placa pode ser facilmente conectada ao computador, proporcionando uma comunicação serial eficiente. A interface também conta com dois **LEDs indicadores** (LED7 e LED8) que sinalizam a transmissão e recepção de dados pela porta serial, facilitando o monitoramento das atividades de comunicação. A representação física da interface USB-UART está ilustrada na **Figura 5**.



Figura 5: Interface física USB-UART, alimentação e LEDs indicadores [4].

A placa também inclui uma interface **VGA**, amplamente utilizada em monitores desde a era dos **CRT** e ainda presente em diversos sistemas. Essa interface, também conhecida como **D-Sub**, utiliza um conector com **15 pinos** organizados em três fileiras, sendo responsável pela transmissão dos sinais de cor **RGB** (vermelho, verde e azul) e dos sinais de sincronização **HSYNC** (horizontal) e **VSYNC** (vertical). Os pinos **1, 2 e 3** transportam os sinais análogicos das cores primárias, variando de **0V** (sem cor) a **0,714V** (cor plena), embora alguns monitores utilizem níveis de até **1Vpp**.

Por fim, a placa AX301 conta com a interface para **SD Card** (*Secure Digital Memory Card*), um tipo de cartão de memória baseado na tecnologia de **flash memory** semicondutora. Desenvolvido em **1999** pela Panasonic, com participação da Toshiba e SanDisk, o SD Card é amplamente utilizado devido ao seu desempenho e custo-benefício, sendo ideal para o armazenamento de dados e configurações em sistemas embarcados.

## 2.5 Periféricos

A placa de desenvolvimento **FPGA AX301** conta com diversos periféricos que facilitam o desenvolvimento e a implementação de projetos digitais. Os **4 LEDs de usuário** permitem a visualização do estado lógico programado no FPGA. Quando a saída do pino do FPGA estiver em **nível lógico 0**, os LEDs permanecem **apagados**. Quando o nível lógico for **1**, os LEDs serão acionados, indicando seu funcionamento de forma simples e intuitiva. A disposição física dos LEDs na placa está representada na **Figura 6**.

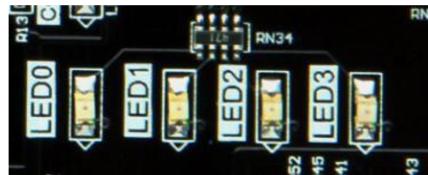


Figura 6: Disposição física dos LEDs de usuário na placa FPGA AX301.

Já o mapeamento dos pinos dos LEDs está apresentado na **Tabela 1**.

Tabela 1: Mapeamento dos pinos dos LEDs na FPGA AX301.

Pin Name	FPGA Pin
LED0	E10
LED1	F9
LED2	C9
LED3	D9

A placa também possui **4 botões independentes**, sendo **3 botões de usuário** (KEY1 a KEY3) e **1 botão de função** (RESET). Quando pressionados, os botões geram um sinal em **nível lógico baixo (0)**, retornando ao **nível lógico alto (1)** quando liberados. Esses botões são amplamente utilizados para controle manual, como reset de sistemas ou acionamento de funções específicas. A disposição física dos botões é exibida na **Figura 7** e o mapeamento de pinos está detalhado na **Tabela 2**.



Figura 7: Disposição física dos botões de usuário e botão RESET na placa FPGA AX301.

Tabela 2: Mapeamento dos pinos dos botões na FPGA AX301.

Key Name	FPGA Pin	Key Number
RESET	N13	RESET
KEY1	M15	KEY 1
KEY2	M16	KEY 2
KEY3	E16	KEY 3

O circuito de cristal ativo de 50 MHz (50M Active Crystal) funciona como a fonte de clock para a placa de desenvolvimento. O sinal de saída do cristal está conectado ao pino global de entrada de **clock da FPGA** (CLK1, pino E1), podendo ser utilizado para acionar circuitos lógicos do usuário dentro da FPGA. Além disso, o usuário pode configurar o PLL (Phase Locked Loop) interno da FPGA para dividir ou multiplicar esse sinal de clock, permitindo a geração de frequências diferenciadas para atender às necessidades do projeto.



Figura 8: Cristal ativo de 50MHz da placa FPGA

Tabela 3: Mapeamento do pino do CLK na FPGA.

Pin Name	FPGA Pin
CLK	E1

O **buzzer** da placa FPGA AX301 é controlado por um **transistor**. Quando o nível lógico enviado pelo FPGA é **baixo (0)**, o transistor é acionado, permitindo a passagem de corrente e fazendo com que o buzzer emita som. Por outro lado, quando o nível é **alto (1)**, o transistor é desligado e o buzzer permanece silenciado.

Para facilitar o controle, existe um **jumper** (CB1) entre o buzzer e o FPGA, que pode ser removido caso o usuário deseje desativar o buzzer de forma permanente. A disposição física do buzzer e seu controle estão ilustrados na **Figura 9**.



Figura 9: Disposição física do buzzer e controle via transistor na FPGA AX301.

Tabela 4: Mapeamento do pino do buzzer na FPGA.

Pin Name	FPGA Pin
Buzzer	C11

O **display de oito segmentos** integrado à placa permite a exibição simultânea de até **seis dígitos**. Esse display é amplamente utilizado em circuitos digitais para representar números e símbolos de maneira visual. A estrutura segmentada do display digital está ilustrada na **Figura 10**, enquanto o mapeamento dos segmentos e a seleção dos dígitos estão detalhados na **Tabela 5**.



Figura 10: Estrutura segmentada do display digital de oito segmentos.

Tabela 5: Mapeamento dos pinos do display digital na FPGA AX301.

Pin Name	FPGA Pin	Descrição
DIG[0]	R14	Segmento A
DIG[1]	N16	Segmento B
DIG[2]	P16	Segmento C
DIG[3]	T15	Segmento D
DIG[4]	P15	Segmento E
DIG[5]	N12	Segmento F
DIG[6]	N15	Segmento G
DIG[7]	R16	Segmento DP (Ponto Decimal)
SEL[0]	N9	Primeiro dígito à direita
SEL[1]	P9	Segundo dígito à direita
SEL[2]	M10	Terceiro dígito à direita
SEL[3]	N11	Quarto dígito à direita
SEL[4]	P11	Quinto dígito à direita
SEL[5]	M11	Sexto dígito à direita

A placa de desenvolvimento AX301 inclui uma **interface de câmera CMOS de 18 pinos**, que pode ser conectada ao módulo de câmera **OV5640** para a implementação da função de captura de vídeo. Após a captura, as imagens podem ser exibidas em um monitor por meio de uma tela **TFT LCD** ou através da interface **VGA**. A escolha do módulo de câmera é flexível, permitindo que os usuários selezionem a opção mais adequada às suas necessidades específicas.

## 2.6 Portas de Expansão

A placa de desenvolvimento FPGA AX301 possui **2 portas de expansão** (J1 e J2), cada uma com **40 sinais**, sendo **1 canal de alimentação de 5V**, **2 canais de alimentação de 3,3V**, **3 canais de aterramento (GND)** e **34 pinos de I/O**. Esses pinos I/O são independentes e não são multiplexados com outros dispositivos, permitindo uma comunicação direta com o FPGA. As portas operam em **nível lógico de 3,3V**, portanto, conectar dispositivos de 5V diretamente pode danificar o FPGA. Para uso com equipamentos de 5V, é necessário utilizar um **chip de conversão de nível**. Para proteção adicional, um resistor de **33 ohms** é conectado em série entre as portas de expansão e os pinos do FPGA, protegendo o sistema contra tensões ou correntes externas. A disposição física das portas de expansão J1 e J2 está ilustrada na **Figura 11**.



Figura 11: Disposição física das portas de expansão J1 e J2 na placa AX301.

As **Tabelas 6** e **7** apresentam o **mapeamento completo dos pinos dos conectores de expansão J1 e J2**, indicando as conexões com os respectivos pinos do **FPGA**.

Tabela 6: Mapeamento dos pinos do conector de expansão J1.

Pin Number	FPGA Pin	Pin Number	FPGA Pin
1	GND	2	VCC5V
3	T14	4	R13
5	T13	6	R12
7	T12	8	R11
9	T11	10	R10
11	T10	12	R9
13	T9	14	R8
15	T8	16	R7
17	T7	18	R6
19	T6	20	R5
21	T5	22	R4
23	T4	24	R3
25	T3	26	P3
27	T2	28	M9
29	L10	30	L9
31	K9	32	P8
33	R1	34	P2
35	P1	36	N2
37	GND	38	GND
39	D3V3	40	D3V3

### 3 Instalação do Software Quartus II

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 12**. Abaixo estão os passos organizados para a instalação e configuração do software.

#### 3.1 Download do Software

- Acesse o site oficial da **Intel** através do link: [Intel Quartus II Web Edition](#).
- Faça o download do arquivo compactado no formato **.tar**.

Tabela 7: Mapeamento dos pinos do conector de expansão J2.

Pin Number	FPGA Pin	Pin Number	FPGA Pin
1	GND	2	VCC5V
3	B1	4	C2
5	B3	6	A2
7	B4	8	A3
9	B5	10	A4
11	B6	12	A5
13	B7	14	A6
15	B8	16	A7
17	B9	18	A8
19	B10	20	A9
21	B11	22	A10
23	B12	24	A11
25	B13	26	A12
27	D5	28	A13
29	C6	30	D6
31	F8	32	E7
33	D8	34	C8
35	E9	36	E8
37	GND	38	GND
39	D3V3	40	D3V3

- Após o download, descompacte o arquivo em um diretório de sua escolha.
- Execute o instalador clicando duas vezes no arquivo setup.

### 3.2 Configuração da Instalação

1. Após abrir o instalador, clique em **Next** para iniciar o processo de instalação.
2. Leia e aceite os termos do contrato selecionando a opção **I accept the agreement** e clique em **Next**.
3. Escolha o diretório onde o software será instalado (ou mantenha o diretório padrão) e clique em **Next**.

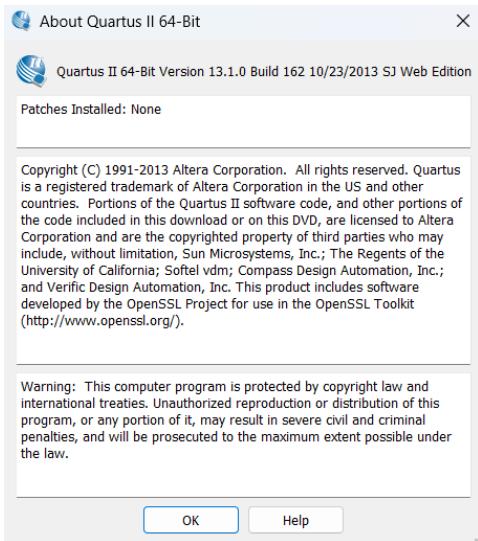


Figura 12: Detalhes da versão do Quartus II utilizada neste guia.

4. Certifique-se de que as opções padrão estão corretamente marcadas, como suporte à versão **Web Edition**, e clique novamente em **Next**.

### 3.3 Conclusão da Instalação

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

## 4 Desenvolvimento de Projeto no Quartus II

Para iniciar um novo projeto no software **Quartus II**, siga o procedimento de criação utilizando o assistente de projetos. Ao abrir o Quartus II, será exibida a tela inicial do ambiente de desenvolvimento, conforme mostrado na **Figura 14**. Nesta tela, clique na opção "**New Project Wizard**" para iniciar o processo de criação de um projeto.

As subseções seguintes abordam detalhadamente o desenvolvimento de circuitos no Quartus II, iniciando com um projeto exemplo clássico de eletrônica digital até a implementação prática, incluindo a criação, escrita e compilação de código VHDL, mapeamento de pinos e o embarque do código no FPGA.

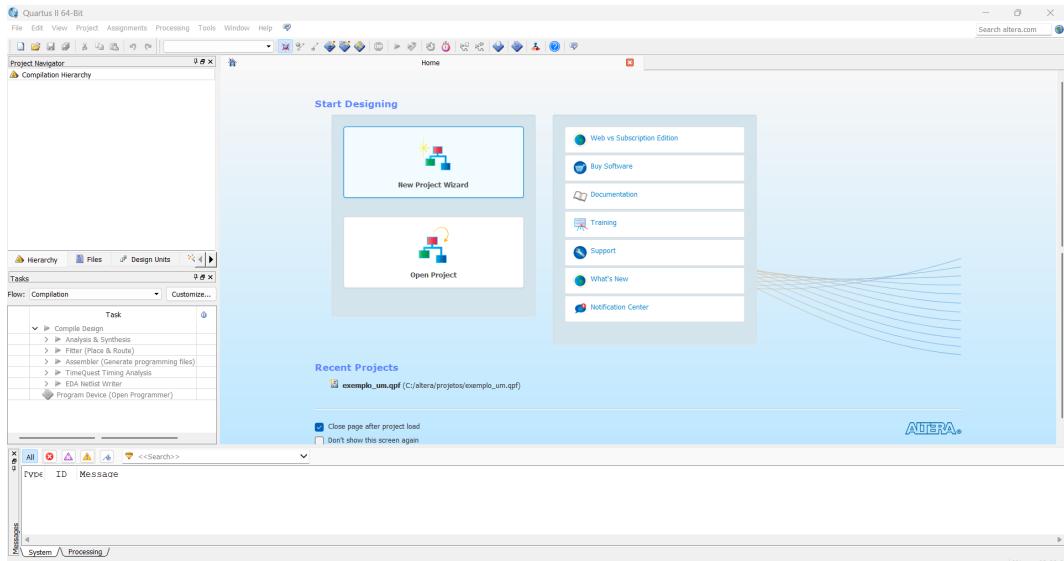


Figura 13: Tela inicial do Quartus II e opção de criação de um novo projeto.

## 4.1 Projeto Detector de Paridade

O **Detector de Paridade** é uma aplicação clássica em eletrônica digital, amplamente utilizada em sistemas digitais para verificar a integridade de dados. Este projeto utiliza os botões da placa FPGA AX301 como entradas binárias e os LEDs como indicadores de paridade. O objetivo é implementar um detector de paridade que verifica se o número de bits "1" em uma palavra de entrada é **par** ou **ímpar**, exibindo o resultado em dois LEDs: **LED0**, que indica paridade par, e **LED1**, que indica paridade ímpar.

### Descrição do Projeto

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

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

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

### Resultados Esperados

A tabela abaixo apresenta o comportamento esperado dos LEDs para diferentes combinações de entrada:

Tabela 8: Comportamento esperado do Detector de Paridade.

Entrada (Botões)	LED0 (Paridade Par)	LED1 (Paridade Ímpar)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

Este projeto combina conceitos fundamentais de lógica digital, como operações XOR e manipulação de bits, proporcionando uma aplicação prática e interativa para estudantes que estão iniciando suas práticas com circuitos digitais.

## 4.2 Criação de um Novo Projeto

1. Abra o Quartus II e clique em **New Project Wizard**.
2. Defina o diretório e o nome do projeto.
3. Configure a família (*Cyclone IV*) e o modelo do FPGA (*EP4CE6F17C8*).
4. Finalize o assistente clicando em **Finish**.

## 4.3 Criação do Arquivo VHDL

1. No menu superior, 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 abaixo implementa o circuito clássico utilizado em sistemas digitais para verificar se o número de bits em nível lógico alto (1) em uma palavra binária é par ou ímpar, em outras palavras o código VHDL do **Detector de Paridade**.

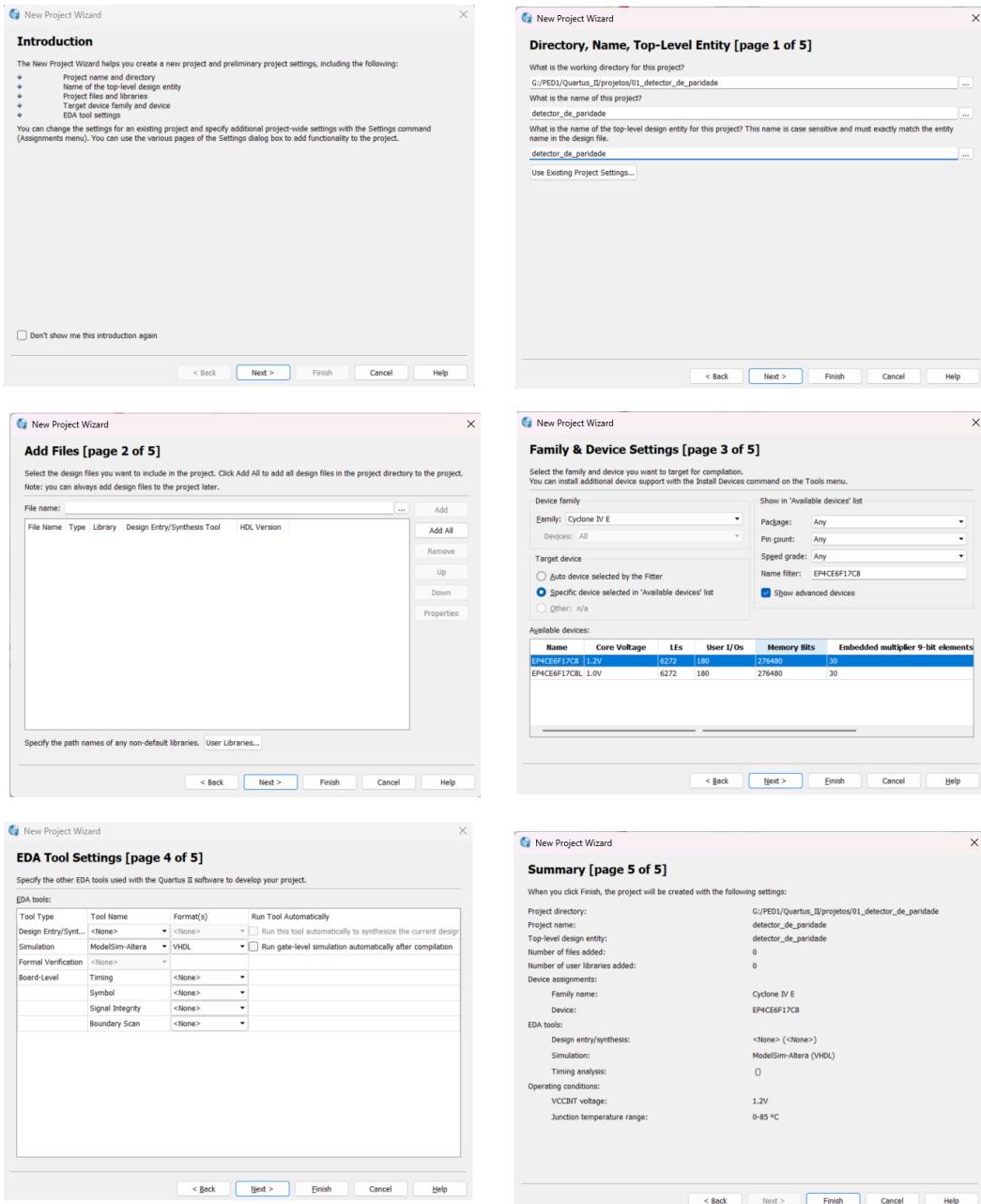


Figura 14: O assistente de criação de projetos no Quartus II guia o usuário em seis etapas: **Introdução, Configuração do Diretório e Nome** (Página 1), **Adição de Arquivos** (Página 2), **Configuração do FPGA** (Página 3), **Ferramentas EDA** (Página 4) e **Resumo Final** (Página 5), facilitando a definição do projeto.

```

1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 ENTITY detector_de_paridade IS
5     PORT (
6         key1 : IN STD_LOGIC;      -- Entrada do botão KEY1
7         key2 : IN STD_LOGIC;      -- Entrada do botão KEY2
8         key3 : IN STD_LOGIC;      -- Entrada do botão KEY3
9         led0 : OUT STD_LOGIC;     -- Saída do LED0 (Paridade Par)
10        led1 : OUT STD_LOGIC;     -- Saída do LED1 (Paridade Ímpar)
11    );
12 END detector_de_paridade;
13
14 ARCHITECTURE behavior OF detector_de_paridade IS
15     SIGNAL paridade : STD_LOGIC;  -- Sinal intermediário
16 BEGIN
17     PROCESS (key1, key2, key3)
18     BEGIN
19         -- Calcular paridade usando XOR
20         paridade <= key1 XOR key2 XOR key3;
21         -- Atribuir resultados aos LEDs
22         led0 <= NOT paridade;   -- LED0 para paridade par
23         led1 <= paridade;       -- LED1 para paridade ímpar
24     END PROCESS;
25 END behavior;
26

```

## 4.4 Compilação do Código VHDL

- Clique em **Start Compilation** para verificar a sintaxe do código.
- Em caso de sucesso, as mensagens na janela **Task** ficarão verdes.

## 4.5 Visualização no RTL Viewer

Após concluir a compilação do projeto no Quartus II, o **RTL Viewer** pode ser utilizado para inspecionar a representação lógica do circuito descrito no código VHDL. Essa ferramenta é essencial para validar a estrutura hierárquica

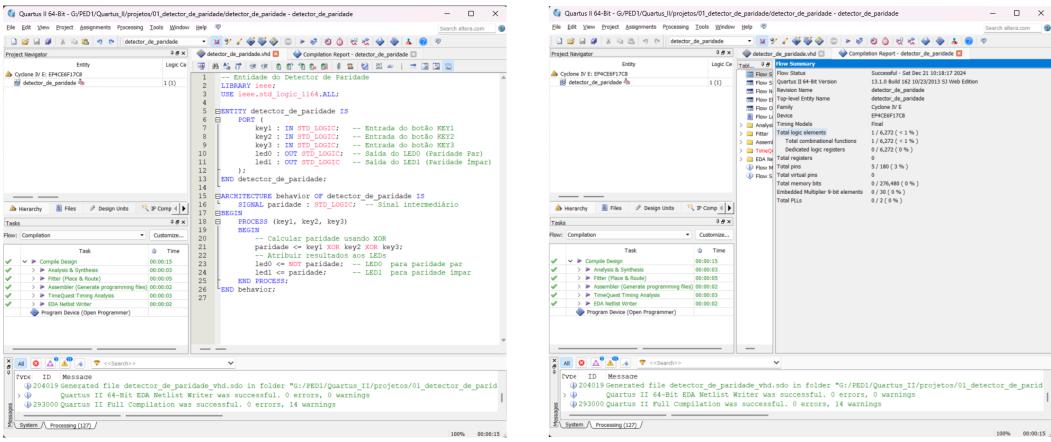


Figura 15: Processo de compilação do projeto Detector de Paridade. A imagem à esquerda mostra a conclusão bem-sucedida de todas as etapas de compilação no Quartus II. À direita, o *Flow Summary* apresenta os detalhes técnicos da compilação, como o uso de elementos lógicos e pinos do FPGA.

e as interconexões do design.

## Passos para acessar o RTL Viewer

1. Acesse o menu superior em **Tools → Netlist Viewers → RTL Viewer**.
2. Aguarde o carregamento da visualização lógica do circuito.

## Análise do RTL Viewer

- A representação hierárquica do circuito com os módulos e sinais.
- As conexões lógicas entre os componentes internos.

Use as ferramentas de zoom e navegação para explorar detalhes específicos da topologia do circuito.

## Exemplo de Verificação no RTL Viewer

Considere o projeto **Detector de Paridade**. Após a compilação:

- O **RTL Viewer** exibirá as conexões dos botões de entrada (KEY1, KEY2, KEY3) aos LEDs de saída (LED0, LED1).

- Verifique se os sinais intermediários e as operações lógicas (XOR) estão implementados corretamente.

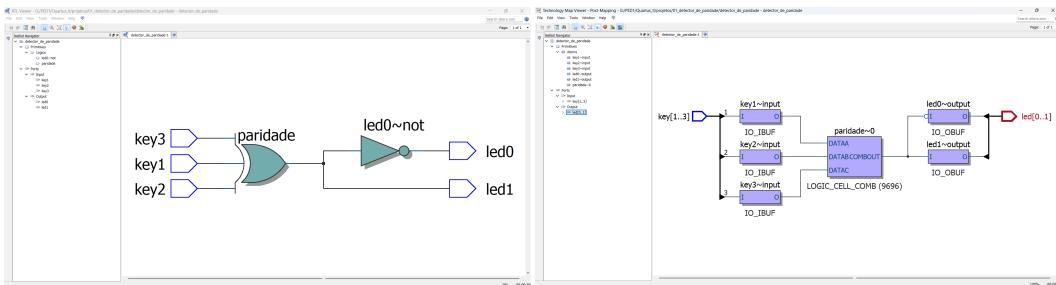


Figura 16: Visualização no RTL Viewer. A imagem à esquerda apresenta a visão lógica do circuito no RTL Viewer, destacando os componentes principais: entradas (key1, key2, key3), a operação lógica XOR (sinal paridade) e as saídas (led0, led1). Já a imagem à direita mostra a visão pós-mapeamento no Technology 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).

## 4.6 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** é uma ferramenta essencial no Quartus II para realizar a configuração dos pinos do FPGA, permitindo associar os sinais lógicos do design às portas físicas do dispositivo. A **Figura 17** apresenta a interface do **Pin Planner**, onde é possível visualizar os pinos disponíveis, atribuir os pinos corretos para as entradas (botões) e saídas (LEDs) do projeto, além de configurar parâmetros adicionais, como força de corrente e taxa de subida.

“

## 4.7 Embarque do Código na FPGA

1. No menu, clique em **Tools → Programmer**.

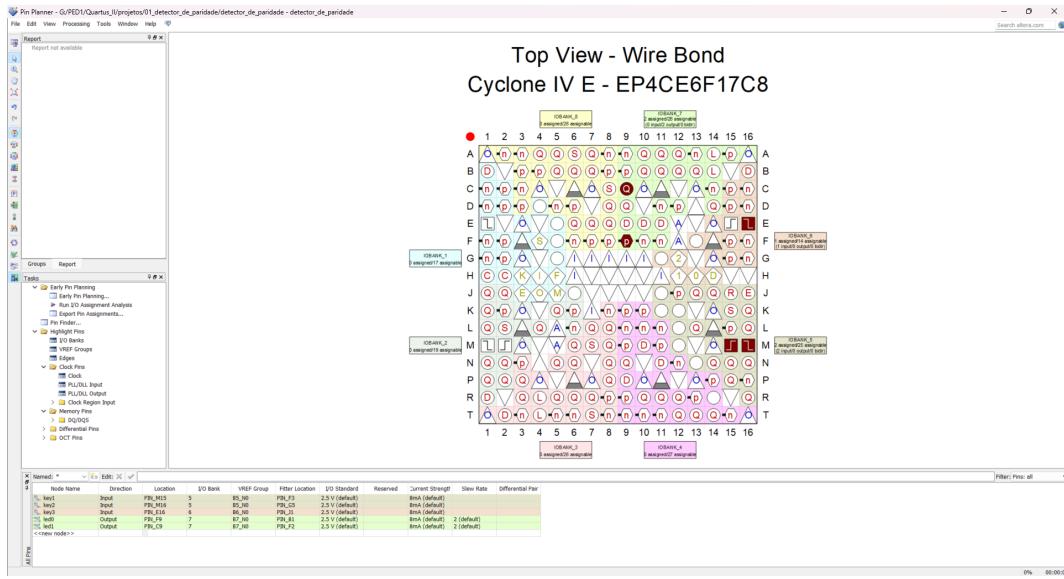


Figura 17: Visualização do *Pin Planner* no Quartus II, mostrando a atribuição de pinos do FPGA Cyclone IV E (EP4CE6F17C8). Nesta tela, é possível configurar as entradas e saídas do design, incluindo os pinos conectados aos botões e LEDs da placa AX301.

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%.

A **Figura 18** ilustra o processo de programação da placa FPGA AX301 utilizando o dispositivo *USB-Blaster*. À esquerda, a imagem mostra a conexão física entre o *USB-Blaster* e a placa FPGA, responsável por transferir o arquivo de configuração gerado no Quartus II para o FPGA.

À direita, a interface do **Quartus II Programmer** exibe a etapa de programação do FPGA, onde o arquivo `.sof` é carregado e configurado no dispositivo. A barra de progresso indica o status da operação, confirmando o sucesso do processo.

## 5 Simulação com ModelSim e GHDL

A simulação do projeto VHDL com o software ModelSim permite validar o comportamento lógico do código antes de embarcá-lo no FPGA. Abaixo estão

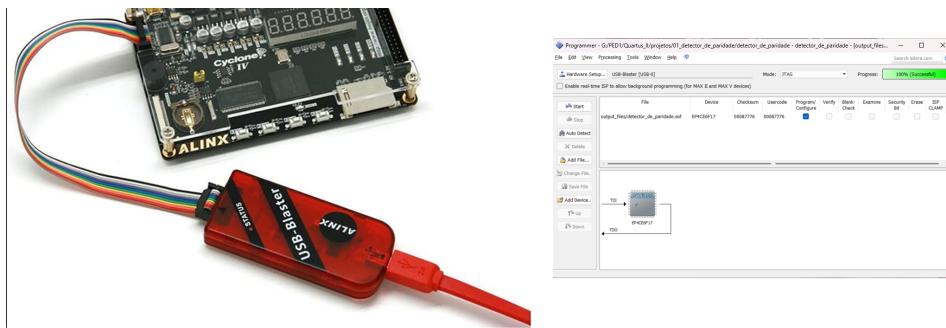


Figura 18: Conexão e programação da placa FPGA AX301 utilizando o *USB-Blaster* e o *Quartus II Programmer*.

os passos organizados para realizar a simulação do **Detector de Paridade**.

## 5.1 Configuração do ModelSim

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

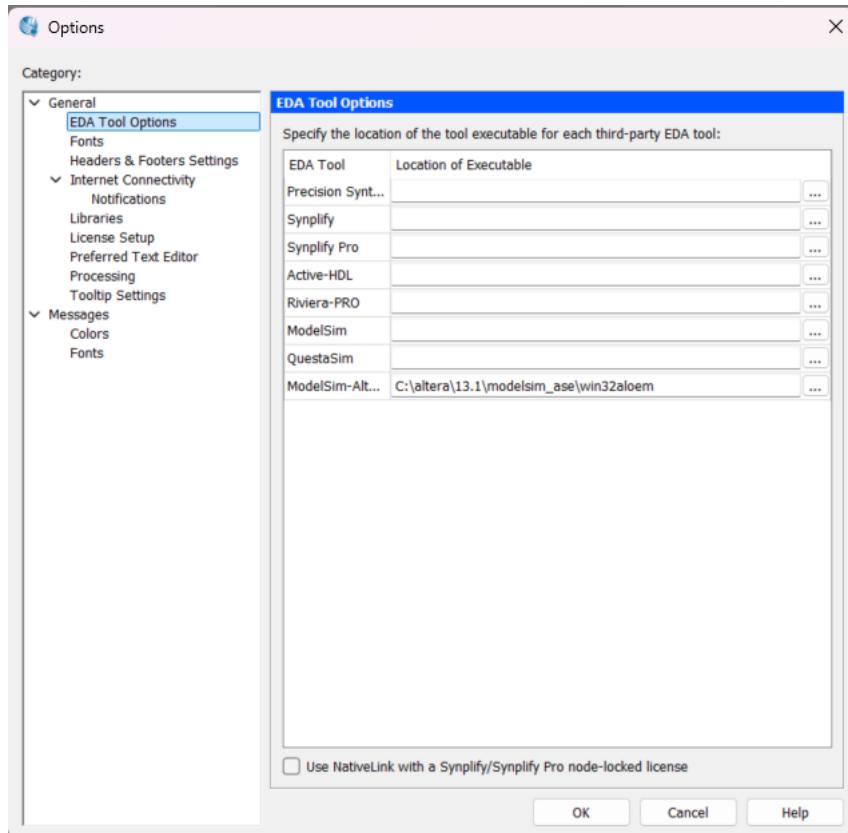


Figura 19: Janela de configurações de ferramentas EDA no Quartus II. Esta imagem mostra as opções de configuração para diferentes ferramentas EDA, incluindo a configuração do caminho para o executável do ModelSim (ModelSim-Altera), que é essencial para realizar simulações dos projetos VHDL. Certifique-se de que o caminho especificado está correto para que o ModelSim funcione adequadamente.

## 5.2 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**.

```

1   -- Testbench para Detector de Paridade
2   LIBRARY ieee;
3   USE ieee.std_logic_1164.ALL;
4
5   ENTITY tb_detector_de_paridade IS
6   END tb_detector_de_paridade;
7
8   ARCHITECTURE behavior OF tb_detector_de_paridade IS
9
10    -- Componentes a serem testados
11    COMPONENT detector_de_paridade
12        PORT (
13            key1 : IN STD_LOGIC;
14            key2 : IN STD_LOGIC;
15            key3 : IN STD_LOGIC;
16            led0 : OUT STD_LOGIC;
17            led1 : OUT STD_LOGIC
18        );
19    END COMPONENT;
20
21    -- Sinais de estímulo
22    SIGNAL key1 : STD_LOGIC := '0';
23    SIGNAL key2 : STD_LOGIC := '0';
24    SIGNAL key3 : STD_LOGIC := '0';
25    SIGNAL led0 : STD_LOGIC;
26    SIGNAL led1 : STD_LOGIC;
27
28 BEGIN
29
30    -- Instância do componente
31    uut: detector_de_paridade PORT MAP (
32        key1 => key1,
33        key2 => key2,
34        key3 => key3,
35        led0 => led0,
36        led1 => led1
37    );
38
39    -- Processo de estímulo

```

```

40  stim_proc: PROCESS
41  BEGIN
42      -- Teste para entrada 000
43      key1 <= '0'; key2 <= '0'; key3 <= '0';
44      WAIT FOR 10 ns;
45
46      -- Teste para entrada 001
47      key1 <= '0'; key2 <= '0'; key3 <= '1';
48      WAIT FOR 10 ns;
49
50      -- Teste para entrada 010
51      key1 <= '0'; key2 <= '1'; key3 <= '0';
52      WAIT FOR 10 ns;
53
54      -- Teste para entrada 011
55      key1 <= '0'; key2 <= '1'; key3 <= '1';
56      WAIT FOR 10 ns;
57
58      -- Teste para entrada 100
59      key1 <= '1'; key2 <= '0'; key3 <= '0';
60      WAIT FOR 10 ns;
61
62      -- Teste para entrada 101
63      key1 <= '1'; key2 <= '0'; key3 <= '1';
64      WAIT FOR 10 ns;
65
66      -- Teste para entrada 110
67      key1 <= '1'; key2 <= '1'; key3 <= '0';
68      WAIT FOR 10 ns;
69
70      -- Teste para entrada 111
71      key1 <= '1'; key2 <= '1'; key3 <= '1';
72      WAIT FOR 10 ns;
73
74      -- Fim da simulação
75      WAIT;
76  END PROCESS;
77
78 END behavior;

```

### 5.3 Execução da Simulação no ModelSim

1. No Quartus II, clique em **Tools → Run Simulation Tool → RTL Simulation**. O ModelSim será aberto automaticamente, carregando o ambiente necessário para a simulação.
2. Na interface do ModelSim, compile o Testbench e o código do projeto para garantir que ambos estão prontos para a simulação:
  - 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**

Esse comando inicializa a simulação e carrega os módulos necessários, permitindo observar os sinais definidos no Testbench.
4. Adicione as formas de onda desejadas à interface gráfica utilizando o comando:
  - **add wave \***

Esse comando adiciona todas as variáveis simuladas à janela de Wave, facilitando a visualização do comportamento do circuito.
5. Execute a simulação para um período de 100 nanosegundos com:
  - **run 100 ns**

Após executar esse comando, as formas de onda dos sinais de entrada e saída serão geradas na janela de simulação, permitindo uma análise detalhada do comportamento lógico do circuito.

A execução dos comandos descritos acima é ilustrada na Figura 20. Na imagem, observa-se, à esquerda, a tela de abertura do ModelSim ALTERA Starter Edition 10.1d, indicando a versão do software utilizada. À direita, é apresentado o ambiente de simulação do ModelSim, onde o console exibe os comandos utilizados para compilar o design principal e o Testbench, carregar a simulação e executar os 100 nanosegundos.

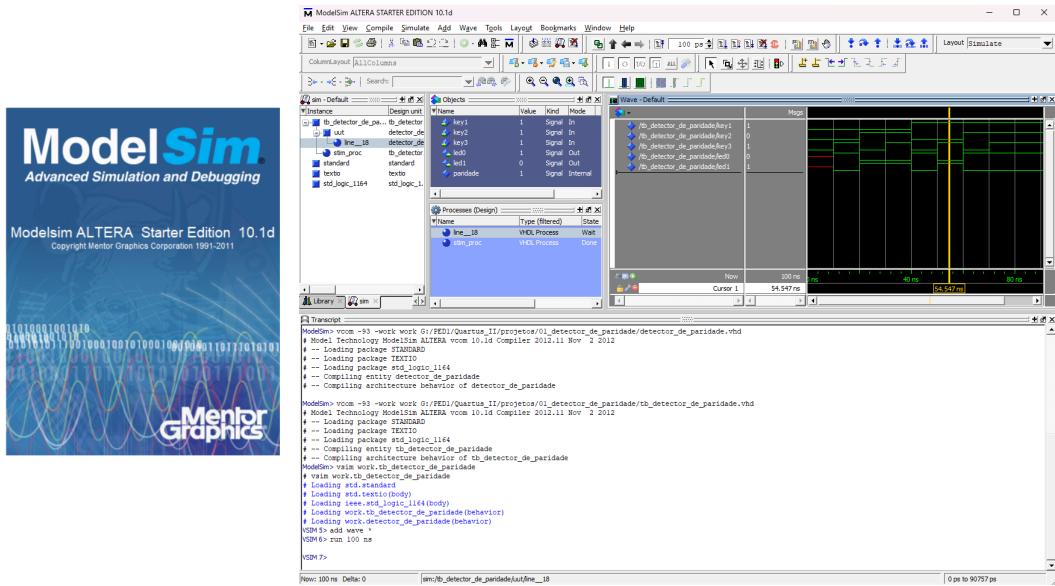


Figura 20: Execução da simulação no ModelSim. A imagem à esquerda mostra a tela de abertura do ModelSim ALTERA Starter Edition 10.1d, indicando a versão do software utilizada. A imagem à direita apresenta o ambiente de simulação com o console exibindo os comandos utilizados para compilar os arquivos VHDL (**detector\_de\_paridade.vhd** e **tb\_detector\_de\_paridade.vhd**), carregar a simulação (**vsim**) e executar (**run 100 ns**). Também são exibidas as formas de onda dos sinais de entrada (**key1**, **key2**, **key3**) e saída (**led0**, **led1**) na janela **Wave**, e informações sobre os processos do Testbench na janela **Processes (Design)**.

## 5.4 Simulação com GHDL e GTKWAVE

O GHDL é uma ferramenta de simulação de código aberto para linguagens VHDL, permitindo a análise, elaboração e simulação de projetos de hardware digital. Ele permite que desenvolvedores validem o funcionamento de circuitos digitais antes de implementá-los em FPGA ou ASIC. Já o GTKWave é uma ferramenta de visualização de formas de onda que permite analisar os sinais gerados durante a simulação. Em conjunto, essas ferramentas são essenciais para o desenvolvimento e validação de sistemas digitais.

Este manual apresenta o uso básico das ferramentas GHDL e GTKWave para simulação e análise de circuitos digitais em VHDL no ambiente Windows. As instruções são executadas via PowerShell.

## 1. Instalação

Certifique-se de que o GHDL e o GTKWave estão corretamente instalados em seu sistema. Para instalação, acesse:

- GHDL: <https://github.com/ghdl/ghdl>
- GTKWave: <http://gtkwave.sourceforge.net/>

## 2. Fluxo de Trabalho

O fluxo de trabalho básico para simulação de um circuito VHDL é composto pelas seguintes etapas:

### (a) Analisar os Arquivos VHDL

Compile os arquivos de descrição do circuito com o comando:

```
ghdl -a detector_de_paridade.vhd  
ghdl -a tb_detector_de_paridade.vhd
```

### (b) Elaborar o Testbench

Realize a elaboração do testbench com o seguinte comando:

```
ghdl -e tb_detector_de_paridade
```

### (c) Executar a Simulação

Execute a simulação e gere o arquivo de forma de onda (.vcd):

```
ghdl -r tb_detector_de_paridade --vcd=onda.vcd
```

### (d) Visualizar as Ondas com GTKWave

Abra o arquivo de forma de onda gerado com o GTKWave para análise gráfica:

```
gtkwave onda.vcd
```

## 3. Exemplo de Execução

Abaixo está um exemplo de execução no PowerShell do Windows:

```
# Análise dos arquivos VHDL  
PS G:\<Caminho>\01_detector_de_paridade> ghdl -a  
    ↳ detector_de_paridade.vhd  
PS G:\<Caminho>\01_detector_de_paridade> ghdl -a  
    ↳ tb_detector_de_paridade.vhd  
  
# Elaboração do Testbench  
PS G:\<Caminho>\01_detector_de_paridade> ghdl -e  
    ↳ tb_detector_de_paridade
```

```

# Execução da Simulação e Geração do Arquivo de Ondas
PS G:\<Caminho>\01_detector_de_paridade> ghdl -r
    ↳ tb_detector_de_paridade --vcd=onda.vcd

# Visualização da Forma de Onda com o GTKWave
PS G:\<Caminho>\01_detector_de_paridade> gtkwave
    ↳ onda.vcd
GTKWave Analyzer v3.3.90 (w)1999-2018 BSI
[0] start time.
[80000000] end time.

```

A Figura 21 apresenta o resultado da simulação do circuito VHDL visualizado no GTKWave. Os sinais de entrada (**key1**, **key2**, **key3**) e de saída (**led0**, **led1**) são exibidos ao longo do tempo, permitindo a análise do comportamento lógico do circuito. Essa visualização facilita a verificação do funcionamento correto do projeto.

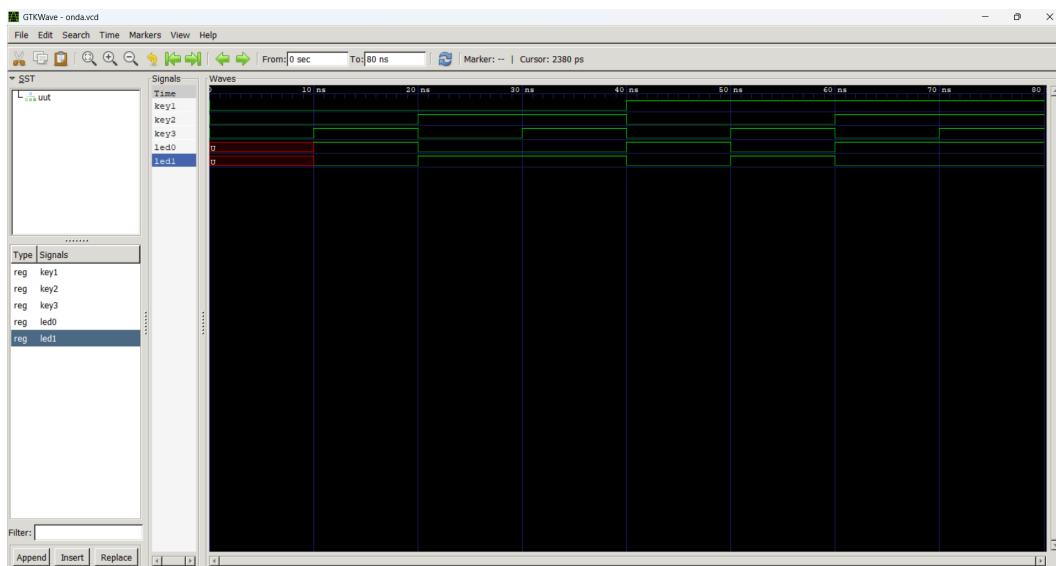


Figura 21: Execução da simulação no GTKWave. A imagem apresenta a análise das formas de onda geradas após a simulação do circuito VHDL. Os sinais de entrada (**key1**, **key2**, **key3**) e de saída (**led0**, **led1**) estão organizados na janela **Signals** à esquerda. Na janela **Waves**, observa-se a evolução temporal dos sinais simulados, permitindo verificar o comportamento lógico do circuito durante a simulação. A simulação cobre um intervalo de 0 a 80 ns.

## 5.5 Análise dos Resultados

- Visualize as formas de onda no **Waveform Viewer**.
- Verifique se os LEDs respondem corretamente às combinações de entrada.
- Confirme que o **led0** acende para paridade par e o **led1** para paridade ímpar.
- Caso os resultados não estejam conforme o esperado, revise o código do Testbench e o projeto principal.

## 6 Considerações Finais

Este manual apresentou os passos essenciais para o uso da placa FPGA AX301 em conjunto com o software Quartus II e ModelSim, incluindo a instalação, configuração e desenvolvimento de um exemplo prático. Ao longo do guia, foram exploradas ferramentas fundamentais para projetos com FPGAs, desde a criação e simulação de designs em VHDL até o embarque do código na placa.

O projeto demonstrado, um detector de paridade, destacou conceitos básicos de eletrônica digital, como operações lógicas e controle de periféricos, sendo uma introdução ideal para estudantes e profissionais que estão iniciando no desenvolvimento com FPGAs.

Os próximos passos recomendados incluem a exploração de funcionalidades mais avançadas da placa AX301, tais como: Integração com sensores externos e displays para aplicações mais interativas; Uso de recursos de memória interna para armazenamento temporário e permanente de dados ou Desenvolvimento de projetos que envolvam interfaces externas, como UART, VGA e SD Card, ampliando a complexidade e as possibilidades de aplicação.

Além disso, o aprendizado e domínio das ferramentas Quartus II e ModelSim abrem caminho para o desenvolvimento de projetos mais desafiadores, como sistemas embarcados e processamento de sinais digitais. Por fim, este manual serve como um ponto de partida para capacitar estudantes e profissionais na criação de projetos com FPGAs, promovendo o aprendizado contínuo e a aplicação prática em diversas áreas de engenharia eletrônica.

## Referências

- [1] Altera Corporation. *Quartus II Handbook*. Intel FPGA, 4th edition, 2023.
- [2] Mentor Graphics Corporation. *ModelSim® User's Manual*. Mentor Graphics Corporation, Wilsonville, Oregon, USA, 2015.
- [3] Departamento de Eletrônica e Computação. *FPGA DE2-115*. Universidade Federal do Amazonas. Manual para utilização da placa DE2-115 com Quartus II 13.0 SP1.
- [4] Rachel Zhou. Fpga development board ax301 - user manual. ALINX, 2020.