

Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Lógica Reconfigurável

Entidade, arquitetura e bibliotecas

Tiago Piovesan Vendruscolo



Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuam o devido crédito aos autores originais. [4.0 international](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Exemplos de HDLs

- HDL – “Hardware description language” ou linguagem de descrição de hardware.

VHDL

- **VHSIC Hardware Description Language**

Verilog

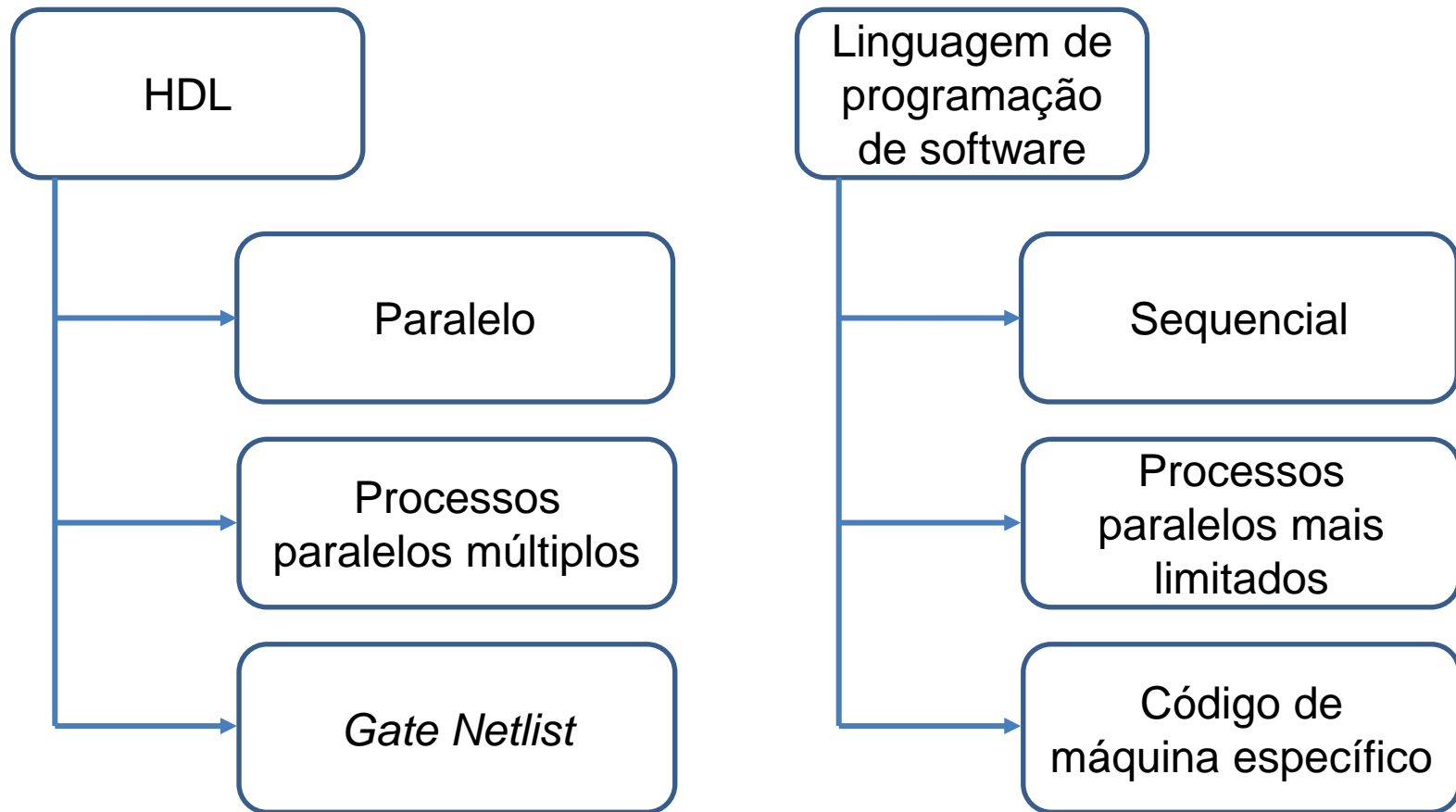
- "verification" and "logic"

AHDL

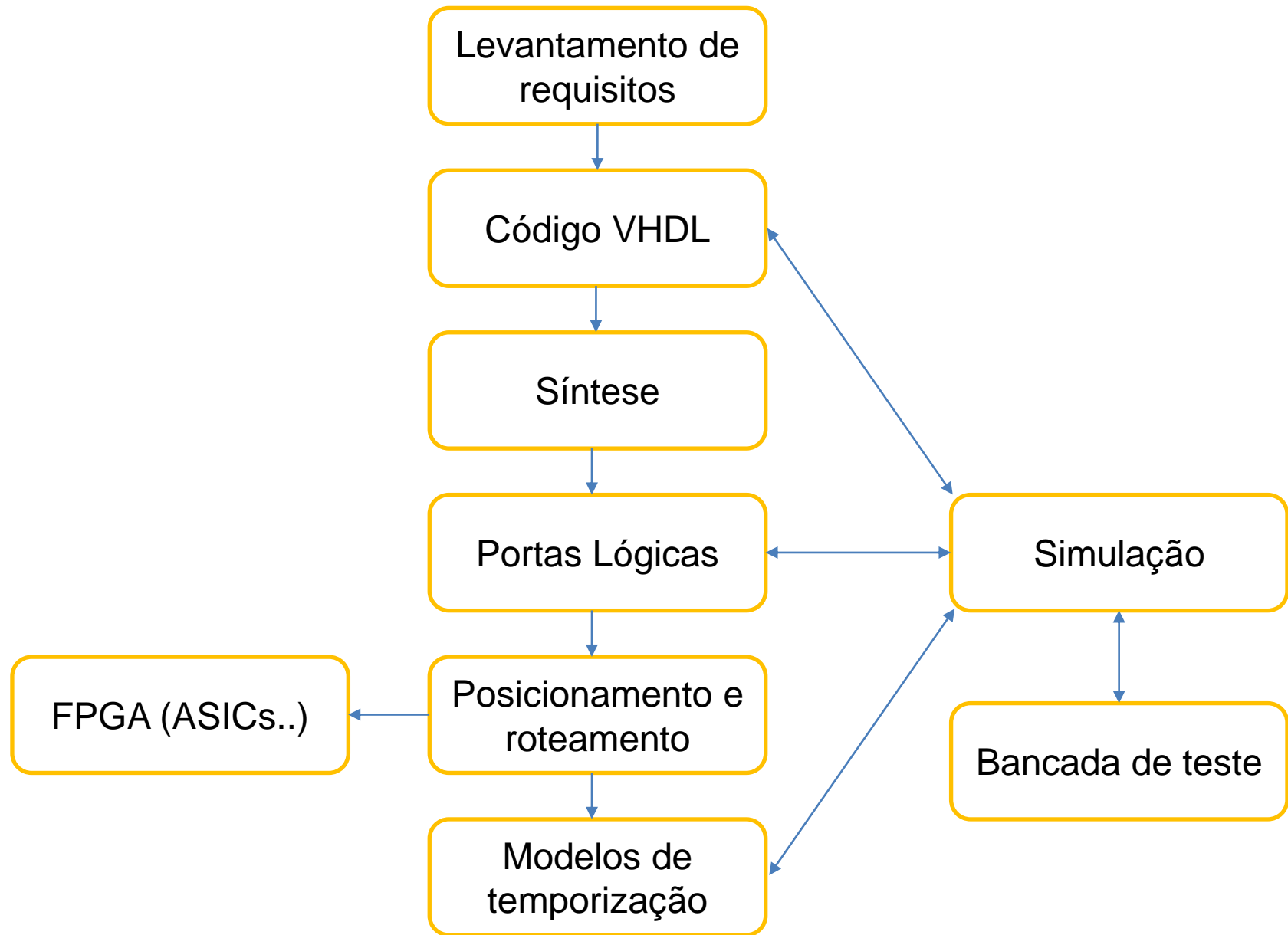
- Altera

O “V” do VHDL: VHSIC "Very High Speed Integrated Circuits"

HDL x Linguagem de programação de software



Fluxo de Projeto



LIBRARY / package declarations

ENTITY

ARCHITECTURE

VHDL – Estrutura de código


```
1  LIBRARY ieee;  
2  USE ieee.std_logic_1164.all;  
3  
4  -----  
5  
6  ENTITY and1 IS  
7  PORT (in1 : IN std_logic;  
8       in2 : IN std_logic;  
9       saida : OUT std_logic);  
10 END and1;  
11  
12 -----  
13  
14 ARCHITECTURE logica OF and1 IS  
15 BEGIN  
16     saida<=IN1 AND IN2;  
17 END logica;
```

LIBRARY / package declarations


ENTITY

ARCHITECTURE

- Contém a lista de todas as bibliotecas e pacotes necessários para o projeto;
- As bibliotecas *std* e *work* já estão incluídas por padrão;

Biblioteca std	
Pacote	Descrição
Standard 	Definições de tipos de dados (BIT, INTEGER, BOOLEAN, etc.) e operadores.
Textio	Utilizada para textos e arquivos.

Biblioteca *ieee*

Pacote	Descrição
<i>std_logic_1164</i> 	Define o tipo de dado de 9 valores STD_ULOGIC e STD_LOGIC, permitindo valores sintetizáveis do tipo <i>don't care</i> ('-') e <i>high-impedance</i> ('Z'). O tipo bit só permite '0' e '1'.
<i>numeric_std</i>	Introduz os tipos SIGNED e UNSIGNED e operadores correspondentes tendo STD_LOGIC como o tipo base.
<i>numeric_bit</i>	O mesmo que o anterior, mas tendo BIT como tipo base.
<i>numeric_std_unsigned</i>	Substituição do pacote não padronizado std_logic_unsigned (VHDL 2008)
<i>numeric_bit_unsigned</i>	Similar ao anterior, mas opera com o tipo BIT_VECTOR ao invés de STD_LOGIC_VECTOR (VHDL 2008)
<i>env</i>	(VHDL 2008) inclui procedimentos de parada e finalização no ambiente de simulação.
<i>fixed_pkg</i>	(VHDL 2008) define os tipos de ponto fixo UFIXED e SFIXED e operadores.
<i>float_pkg</i>	(VHDL 2008) define o tipo de ponto flutuante FLOAT.

Biblioteca <i>ieee</i>	
Pacote	Descrição
<i>std_logic_arith</i>	Define os tipos SIGNED e UNSIGNED e operadores. É parcialmente equivalente ao <i>numeric_std</i> .
<i>std_logic_unsigned</i>	Introduz funções que permitem aritmética, comparação e deslocamento com sinais do tipo STD_LOGIC_VECTOR operando como números sem sinal.
<i>std_logic_signed</i>	Introduz funções que permitem aritmética, comparação e deslocamento com sinais do tipo STD_LOGIC_VECTOR operando como números com sinal.

```
1 -----
2 LIBRARY library_name; --Biblioteca
3 USE library_name.package_name.all; --pacote
4 -----
5 LIBRARY std; --optional declaration
6 USE std.standard.all; --optional declaration
7 LIBRARY work; --optional declaration
8 USE work.all; --optional declaration
9 -----
10 LIBRARY ieee;
11 USE ieee.std_logic_1164.all; --Declaração necessária
12 USE work.my_package.all; --Declaração necessária
13 -----
```

*WORK é utilizado para apontar uma biblioteca localizada no diretório do projeto.

Utilizando apenas uma parte do pacote

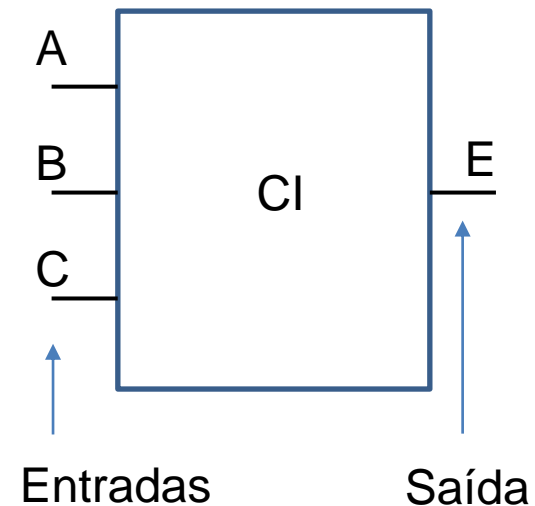
```
-----  
LIBRARY library_name;  
USE library_name.package_name.package_parts;  
-----
```

```
-----  
1 LIBRARY ieee;  
2 USE ieee.std_logic_1164.all;  
3 LIBRARY work;  
4 USE work.componentes_circuito.porta_inversora;  
5 USE work.componentes_circuito.porta_and;  
-----
```

- ENTITY (entidade): Descreve a interface do componente. Especifica os pinos de entrada e saída do circuito.
- O PORT indica as portas de entrada e saída. Todos os elementos dentro do PORT são sinais e não variáveis.
- Uma entidade pode ser pensada como um símbolo para um componente.

```
ENTITY CI IS  
PORT ( A,B,C: IN BIT;  
       E: OUT BIT);  
END ENTITY;
```

- A entidade pode ser finalizada por “END ENTITY” ou “END nome_da_entidade”, no caso acima “END CI”.
- *VHDL não é case sensitive*



- A declaração PORT possui 3 partes:

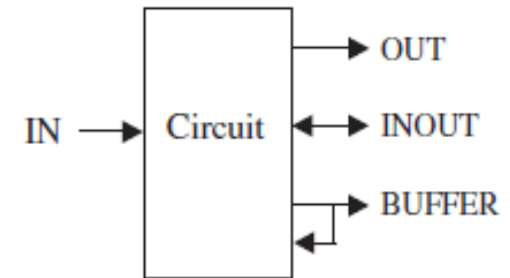
- *Nome, modo e tipos de dados;*

```
ENTITY CI IS  
PORT (<NOME> : <MODO> <TIPOS_DADOS>);  
END ENTITY;
```

```
ENTITY CI IS  
PORT ( A,B,C: IN BIT;  
       E: OUT BIT);  
END ENTITY;
```

- Modos do PORT:

- *IN – unidirecional;*
 - *OUT – unidirecional;*
 - *INOUT – bidirecional;*
 - *BUFFER – Saída que pode ser lida internamente.*



- Apenas letras, dígitos e underline podem ser usados;
- O primeiro caractere deve ser uma letra;
- O último caractere não pode ser underline;
- Não são permitidos dois underline consecutivos.

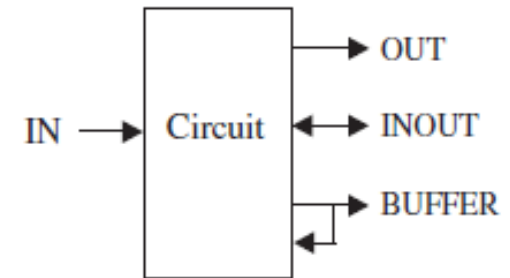
- A declaração PORT possui 3 partes:

- *Nome, modo e tipos de dados;*

```
ENTITY CI IS  
PORT (<NOME> : <MODO> <TIPOS_DADOS>);  
END ENTITY;
```

- Tipo de dados:

- BIT – Valores '0' e '1'.
 - BIT_VECTOR – Vetor de bits
 - BIT_VECTOR (2 downto 0) – (D2 D1 D0)
 - BIT_VECTOR (0 to 2) – (D0 D1 D2)
 - STD_LOGIC – 9 tipos de valores – '0', '1', 'Z', '-' (don't care), ...
 - STD_LOGIC_VECTOR
 - *E muitos outros que veremos nas próximas aulas.*

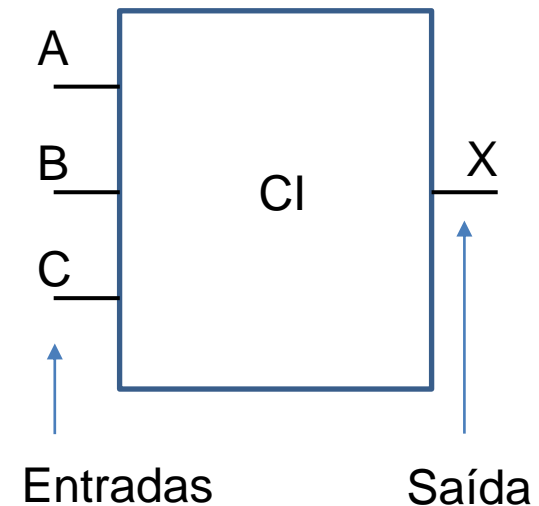


- A declaração PORT possui 3 partes:
 - *Nome, modo e tipos de dados;*

```
ENTITY CI IS  
PORT (<NOME> : <MODO> <TIPOS_DADOS>);  
END ENTITY;
```

Exemplo:

```
ENTITY CI IS  
PORT ( A,B,C: IN STD_LOGIC;  
       X: OUT STD_LOGIC);  
END ENTITY;
```



VHDL – ARCHITECTURE

- ARCHITECTURE: Contém o código VHDL, que descreve como o circuito deve se comportar (funções).
 - *Declarations (opcional): Onde sinais e constantes são declaradas.*

```
ARCHITECTURE architecture_name OF entity_name IS
    [declarations - signal e constant]
BEGIN
    (code)
END architecture_name;
```

`<=` assinala o valor a um sinal e PORT.

`:=` assinala o valor a uma variável e constante.

- Exemplo: Porta AND

```
ARCHITECTURE aula OF porta_and IS
BEGIN
    X <= A AND B;
END aula;
```

Aqui estamos trabalhando apenas com os PORT.

Não é permitido utilizar palavras reservadas em VHDL para nomear Entidade e Arquitetura.

abs	access	after	alias	all	and	architecture
array	assert	attribute	begin	block	body	buffer
bus	case	component	configuration	constant	downto	disconnect
else	elsif	end	entity	exit	file	function
for	generate	generic	guarded	if	in	inout
is	label	library	linkage	loop	map	mod
nand	new	next	nor	not	null	of
on	open	or	others	out	port	package
process	procedure	range	record	register	rem	report
return	select	severity	signal	subtype	then	to
type	transport	units	until	use	variable	wait
when	while	with	xor			

Atribuição de variável e constante:

```
A := 50;
```

Atribuição de sinal e PORT:

```
B <= A AND C;
```

VARIABLE, SIGNAL e CONSTANT são declarados dentro da ARCHITECTURE.
PORTs são declarados dentro da ENTITY.

- Operadores Lógicos:
 - *NOT* (tem prioridade)
 - *AND*
 - *OR*
 - *NAND*
 - *XOR*
 - *XNOR*
- Exemplo:
 - *y <= NOT (a AND b) ;*
 - *y <= a NAND b ;*

- Comparações lógicas:
 - `=` (*igual*)
 - `/=` (*diferente*)
 - `>` (*maior*)
 - `<` (*menor*)
 - `<=` (*menor ou igual*)
 - `>=` (*maior ou igual*)

Os tipos de operando em uma operação relacional devem ser iguais.

- Operadores Aritméticos:
 - $+$ (*soma*)
 - $-$ (*subtração ou negação*)
 - $*$ (*multiplicação*)
 - $/$ (*divisão*)
 - *mod* (*módulo*)
 - *rem* (*resto da divisão*)
 - *abs* (*valor absoluto*)
 - $**$ (*Exponenciação*)

❑ Em códigos concorrentes usa-se:

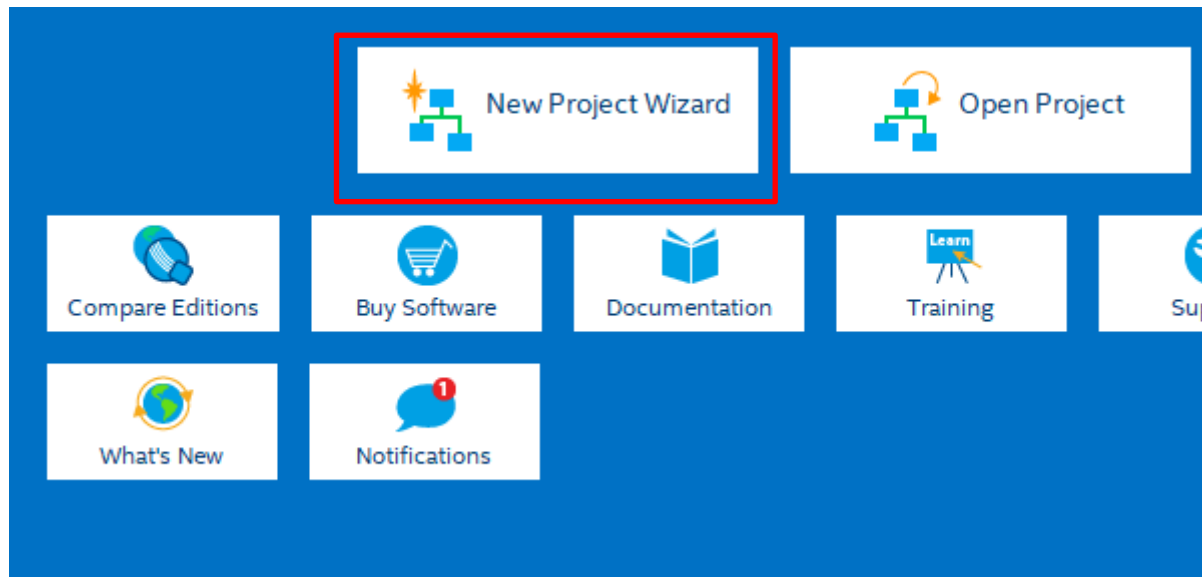
- Operadores;
- WHEN (WHEN/ELSE ou WITH/SELECT/WHEN);
- GENERATE;
- BLOCK;

❑ Em códigos sequenciais usa-se:

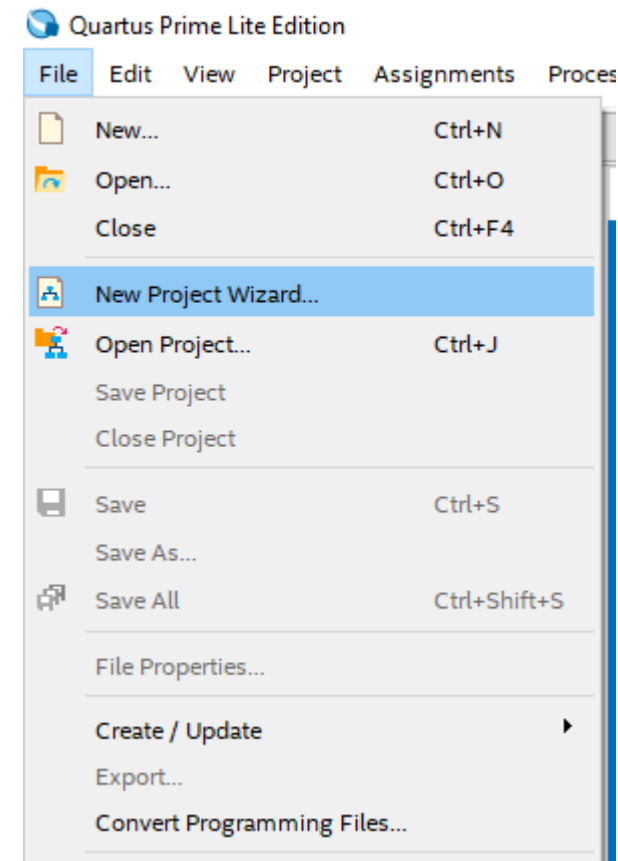
- IF-THEN-ELSE;
- CASE-WHEN;
- FOR-LOOP;
- WHILE-LOOP;
- WAIT (a partir do VHDL 2008).

Criando um projeto no Quartus Prime 18

Pela tela inicial:



Pelo menu File:



Criando um projeto no Quartus

New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:\intelFPGA_lite\18.1

What is the name of this project?

Primeiro_projeto

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.

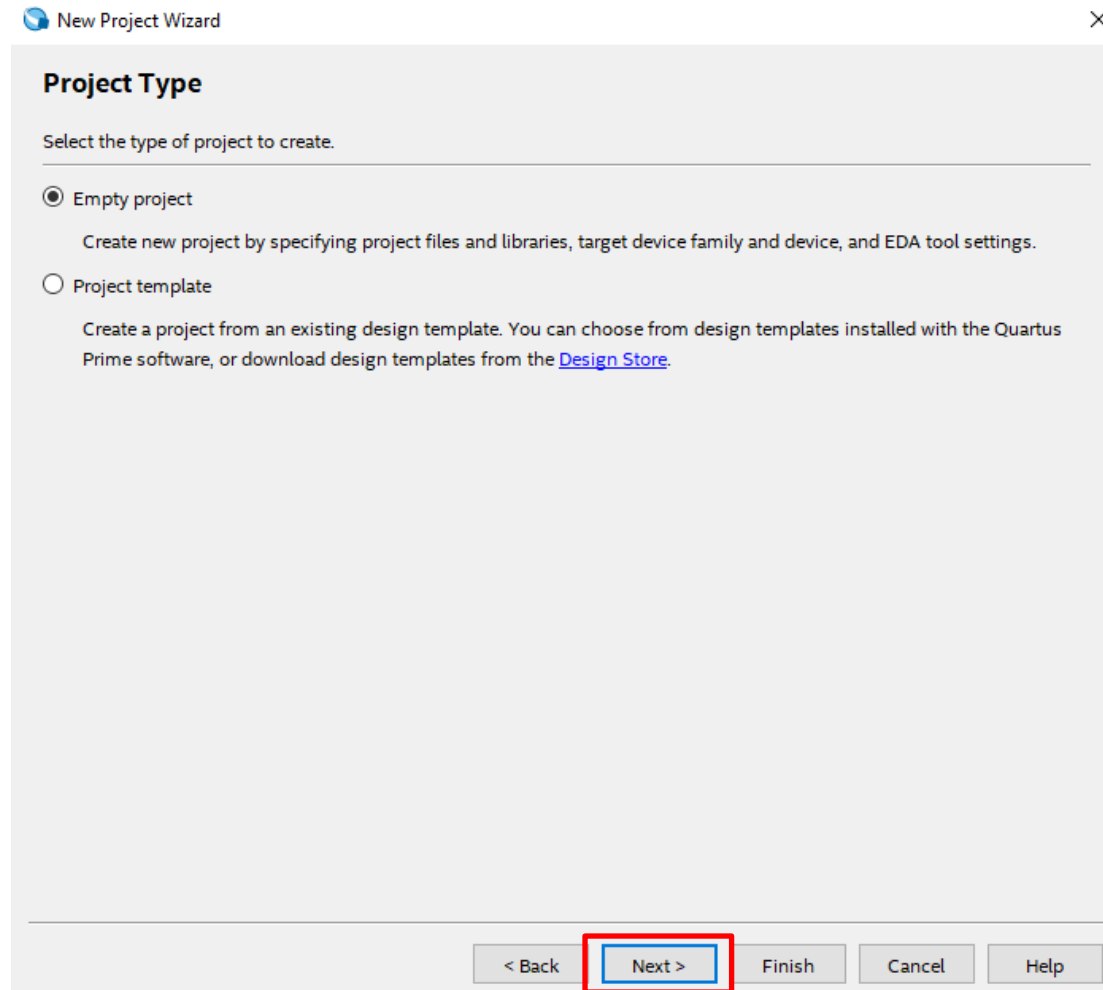
Primeiro_projeto

Use Existing Project Settings...

< Back Next > Finish Cancel Help

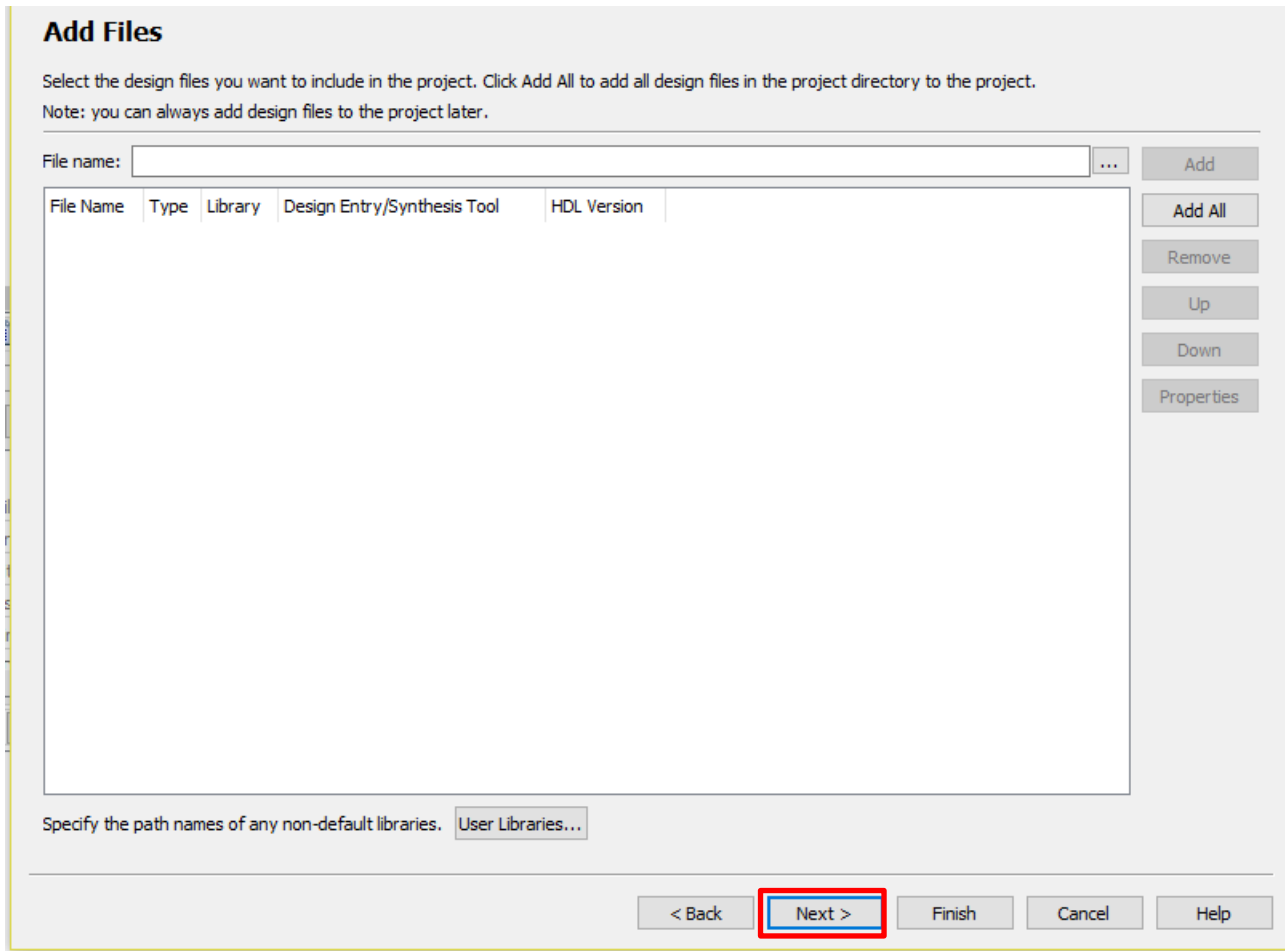
Criando um projeto no Quartus

- Nessa tela é selecionado se o projeto será iniciado do zero ou se usará um template.



Criando um projeto no Quartus

- Nessa parte é possível adicionar arquivos externos que serão utilizados no projeto.



Criando um projeto no Quartus > CYCLONE IV E > DEVICE EP4CE22F17C6

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: Cyclone IV E

Device: All

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:

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bi
EP4CE22F17A7	1.2V	22320	154	154	608256	132
EP4CE22F17C6	1.2V	22320	154	154	608256	132
EP4CE22F17C7	1.2V	22320	154	154	608256	132

< Back Next > Finish Cancel Help

Criando um projeto no Quartus

EDA Tool Settings

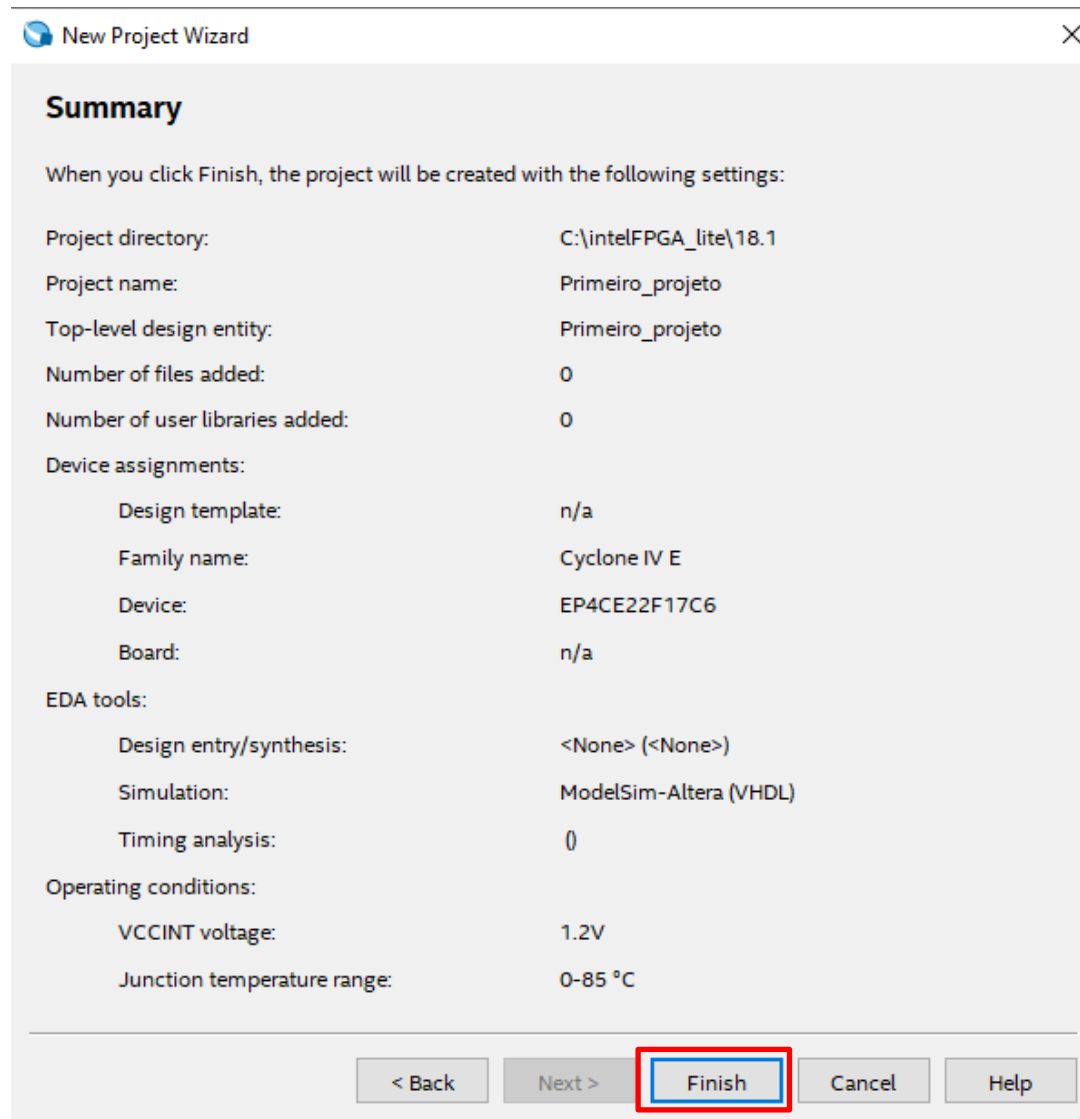
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

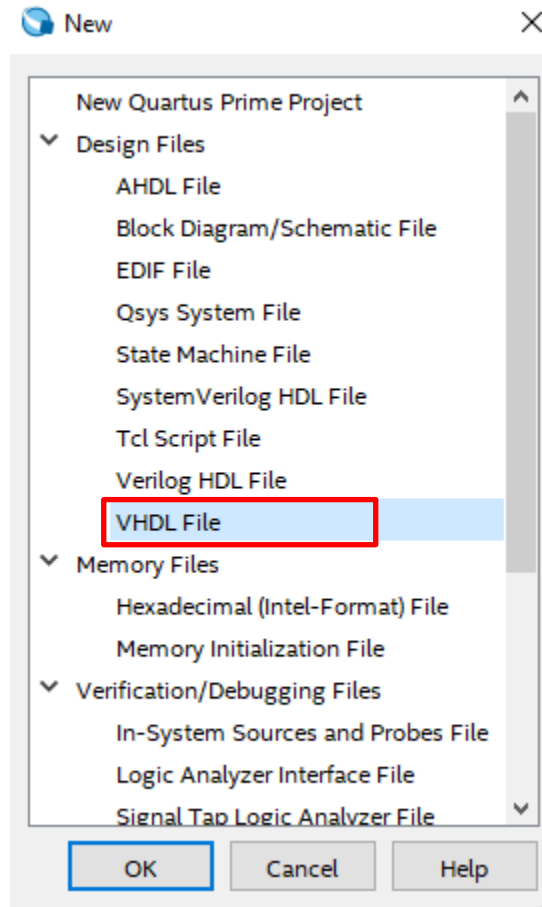
Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synth...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	VHDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level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

< Back **Next >** Finish Cancel Help

Criando um projeto no Quartus



Criando um projeto no Quartus



VHDL – Exemplo 1

- Projetar uma porta AND.

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;
```

```
ENTITY Primeiro_projeto IS  
    PORT (IN1 : IN std_logic;  
          IN2 : IN std_logic;  
          SAIDA : OUT std_logic);  
END Primeiro_projeto;
```

Use o mesmo nome do projeto

```
ARCHITECTURE logica OF Primeiro_projeto IS  
BEGIN  
    SAIDA<=IN1 AND IN2;  
END logica;
```

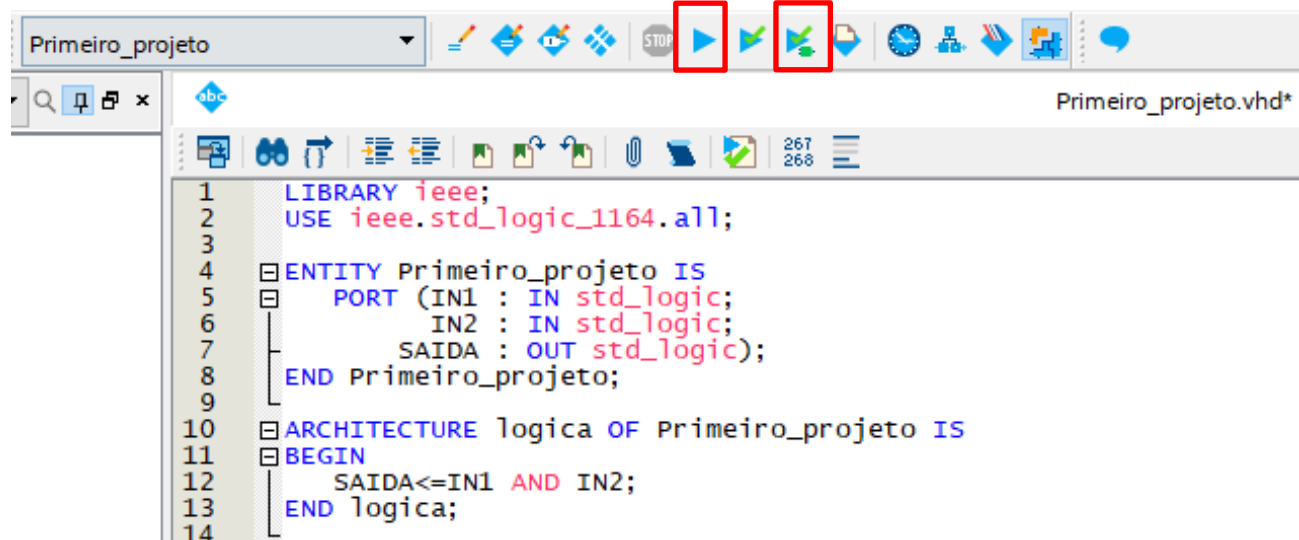
- Salve com o mesmo nome do projeto.

VHDL – Exemplo 1

- Compile o código

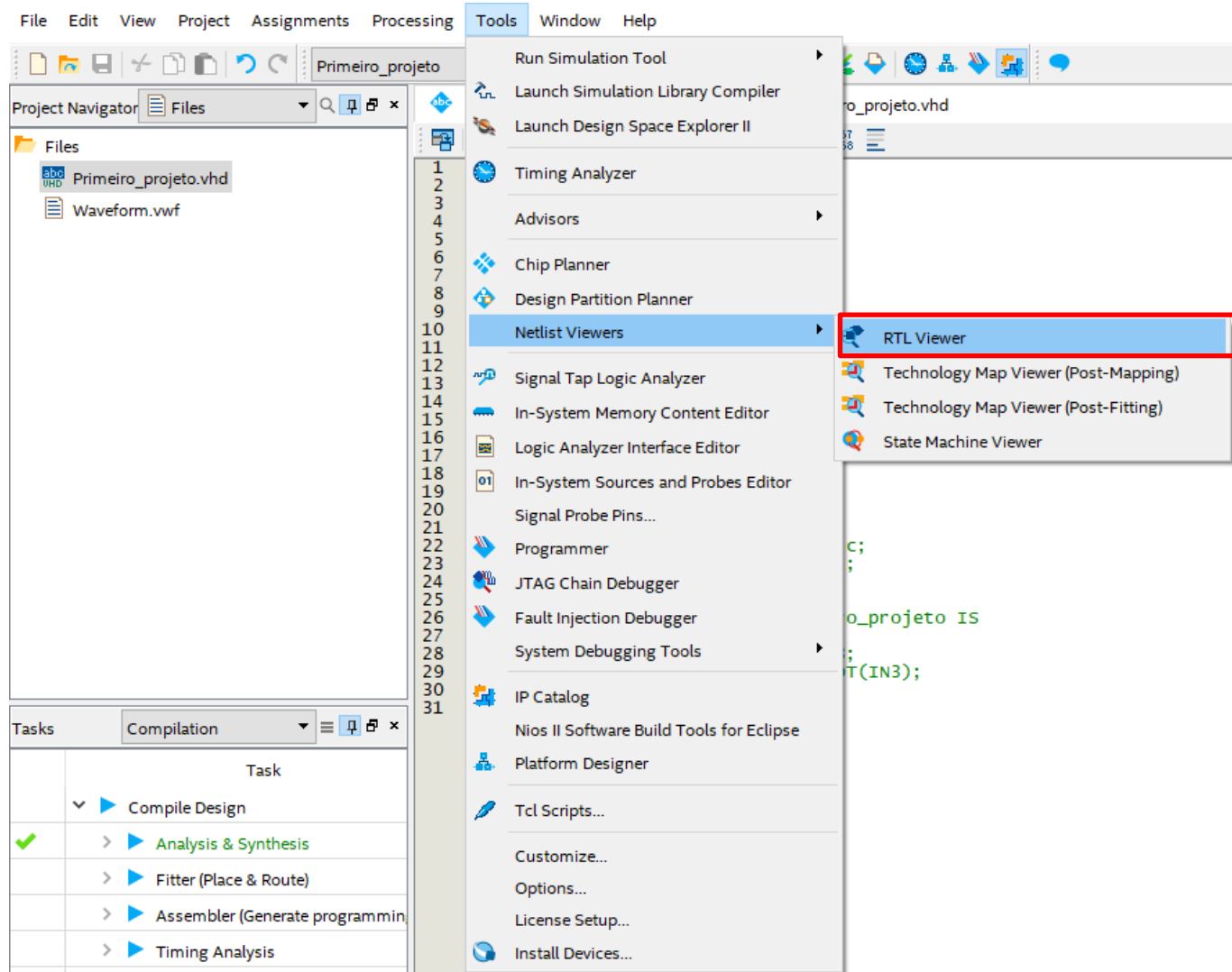
Compile (Além de verificar o código e criar o arquivo de simulação, também gera o arquivo para gravar na FPGA, específico para o modelo selecionado durante a criação do projeto)

Start analysis and synthesis (usado para verificar o código e criar o arquivo de simulação)



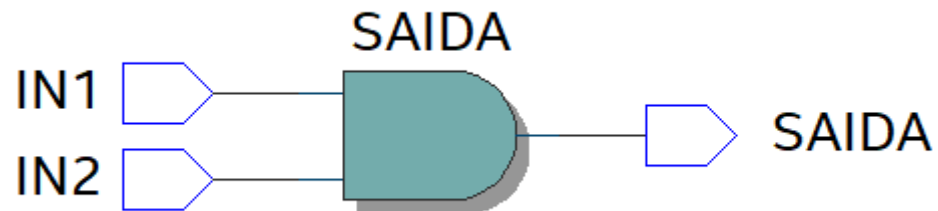
VHDL – Exemplo 1

- Verifique o circuito gerado pelo RTL VIEWER (Register Transfer Level)



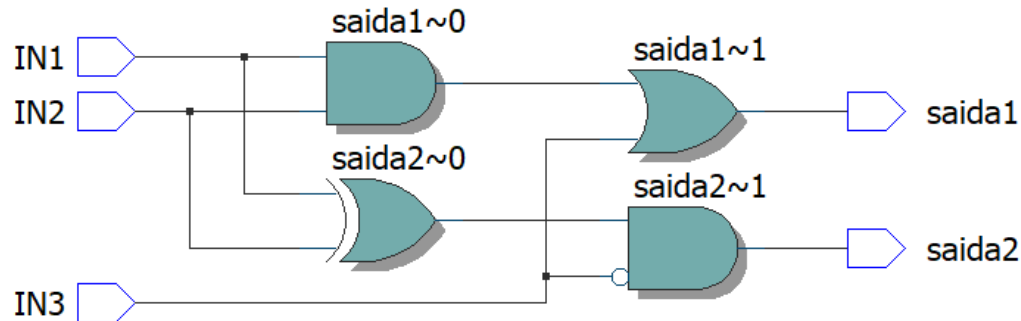
VHDL – Exemplo 1

- Verifique o circuito gerado pelo RTL VIEWER (Register Transfer Level)



VHDL – Exercício 1

- Escreva o código do circuito combinacional abaixo:
- Obs: Utilize "--" para comentar uma linha de código.
 - *Selecione um trecho do código > botão direito > comment selection.*



VHDL – Exercício 1

- Resposta:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Primeiro_projeto IS
  PORT (IN1 : IN std_logic;
        IN2 : IN std_logic;
        IN3 : IN std_logic;
        SAIDA1 : OUT std_logic;
        SAIDA2 : OUT std_logic);
END Primeiro_projeto;

ARCHITECTURE logica OF Primeiro_projeto IS
BEGIN
  SAIDA1<= (IN1 AND IN2) OR IN3;
  SAIDA2<= (IN1 XOR IN2) AND NOT(IN3);
END logica;
```

VHDL – Exercício 1

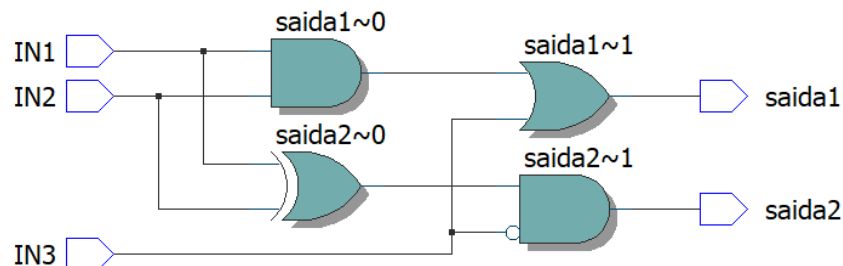
■ Compilation Report:

Analysis and synthesis: Compilação rápida, mostrando a quantidade de elementos lógicos e pinos utilizados.

Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ex1
Top-level Entity Name	ex1
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	2
Total registers	0
Total pins	5
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

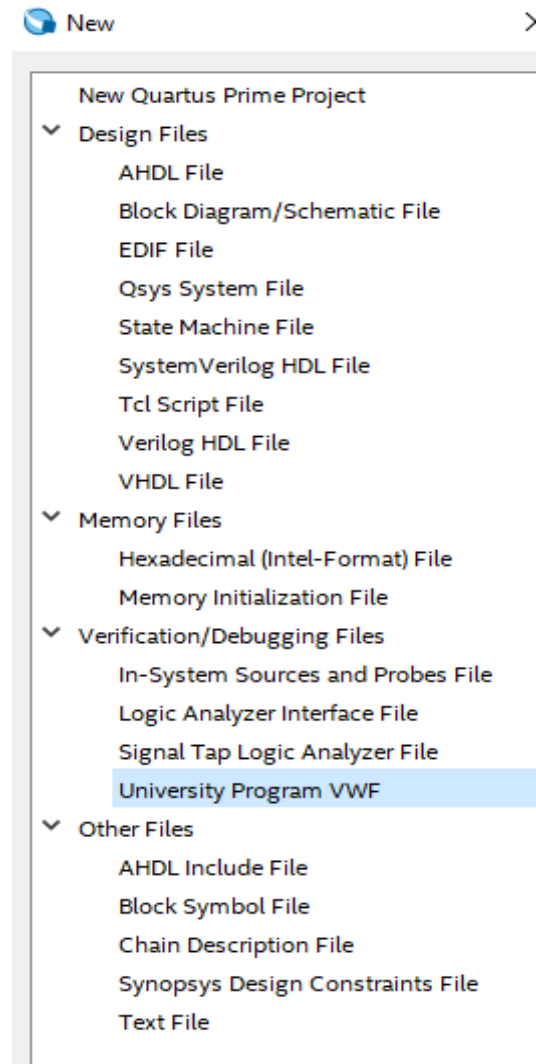
Compile: Compilação lenta, também mostra a porcentagem de recursos utilizados da FPGA.

Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ex1
Top-level Entity Name	ex1
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	2 / 22,320 (< 1 %)
Total registers	0
Total pins	5 / 154 (3 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)



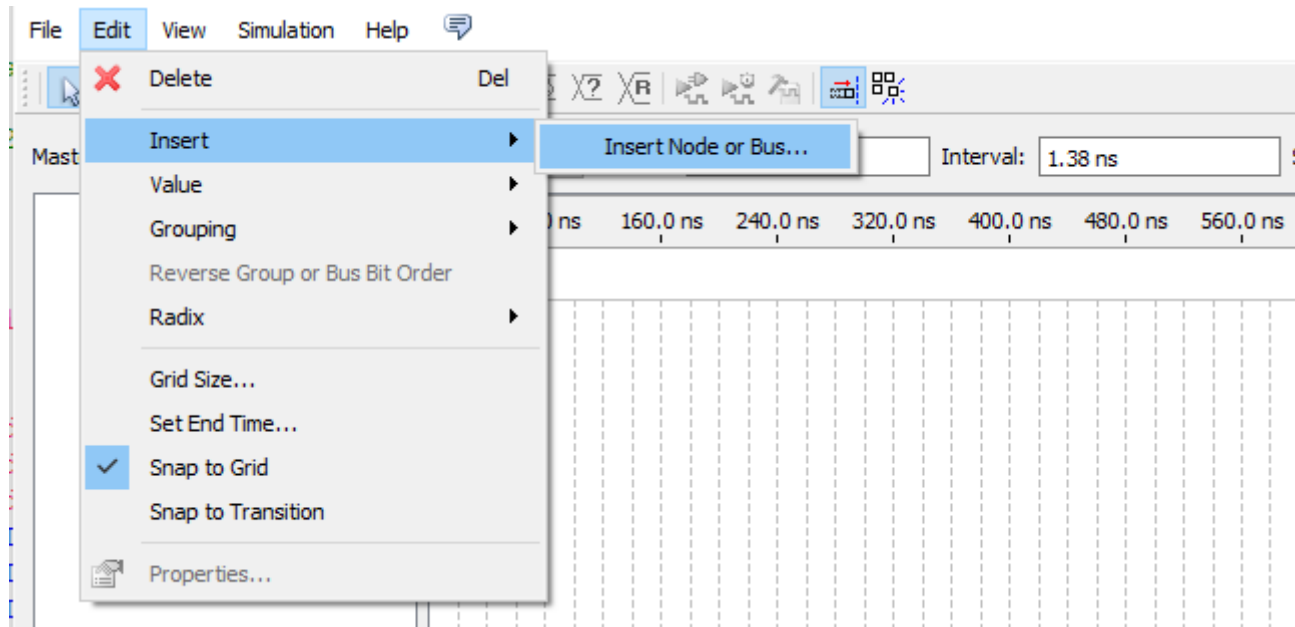
VHDL – Exercício 1

- Simulando o circuito: File > New > University Program VWF



VHDL – Exercício 1

- Simulando o circuito:



VHDL – Exercício 1

Simulando o circuito:

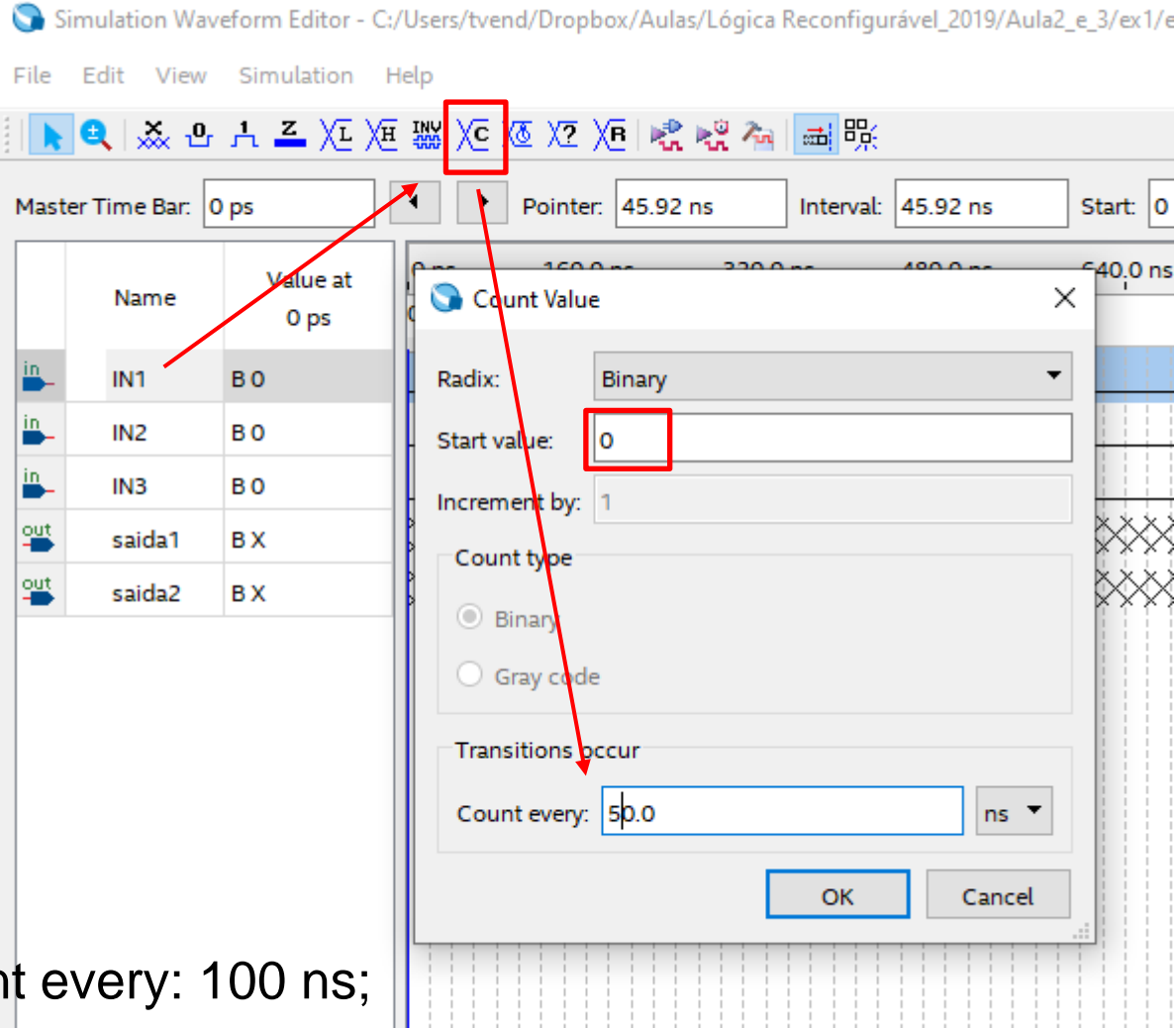
The screenshot shows the Simulation Waveform Editor interface. The main window displays a waveform with a Master Time Bar at 0 ps and a Pointer at 10.6 ns. The waveform has columns for 0 ps, 160.0 ns, 320.0 ns, 480.0 ns, 640.0 ns, 800.0 ns, and 960.0 ns.

Two dialog boxes are open:

- Node Finder**: This dialog is used to find nodes in the circuit. It has a "Named:" field with an asterisk, a "Filter:" dropdown set to "Pins: all", and a "Look in:" field with an asterisk. The "List" button is highlighted with a red box and labeled with a red "2". The "Nodes Found:" table lists nodes: IN1, IN2, IN3 (all Inputs) and saida1, saida2 (both Outputs). The "Selected Nodes:" table is empty. A red box labeled "3" highlights the ">>" button between the two tables. A red arrow labeled "4" points from the "List" button to the "Insert Node or Bus" dialog.
- Insert Node or Bus**: This dialog is used to insert a new node or bus. It has a "Name:" field with the text "Use Node Finder to insert ...", a "Type:" dropdown set to "INPUT", a "Value type:" dropdown set to "9-Level", a "Radix:" dropdown set to "Binary", a "Bus width:" field set to "1", and a "Start index:" field set to "0". The "Node Finder..." button is highlighted with a red box and labeled with a red "1".

VHDL – Exercício 1

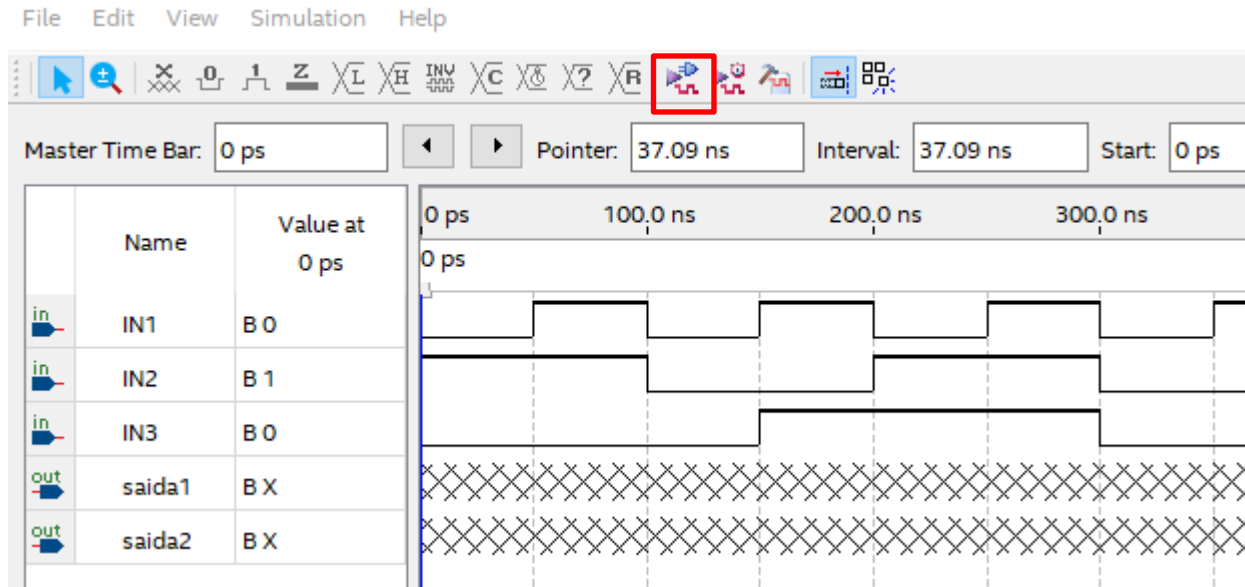
- Edit > Set end time : 500 ns.
- Edit > Grid size : 50 ns.



- IN2: Start value: 1; count every: 100 ns;
- IN3: Start value: 0; count every: 150 ns;

VHDL – Exercício 1

- Simule clicando em run functional simulation.
 - *Aparecerá uma janela para salvar o arquivo de simulação caso ainda não tenha salvo.*



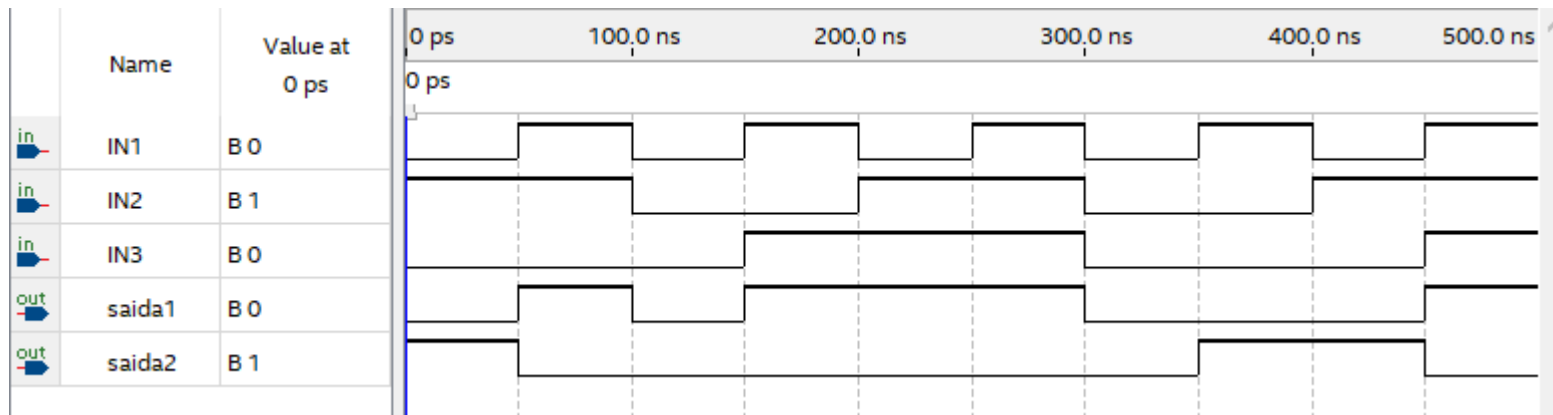
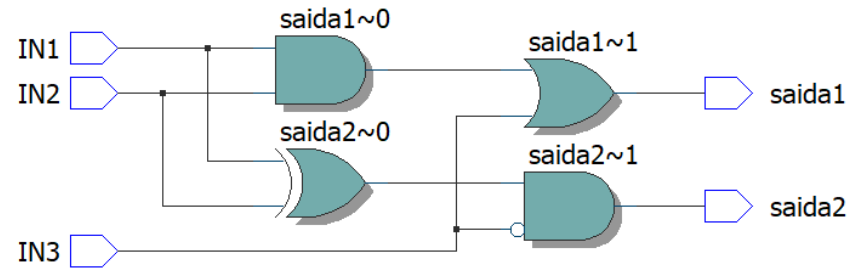
VHDL – Exercício 1

■ Resultado da simulação:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

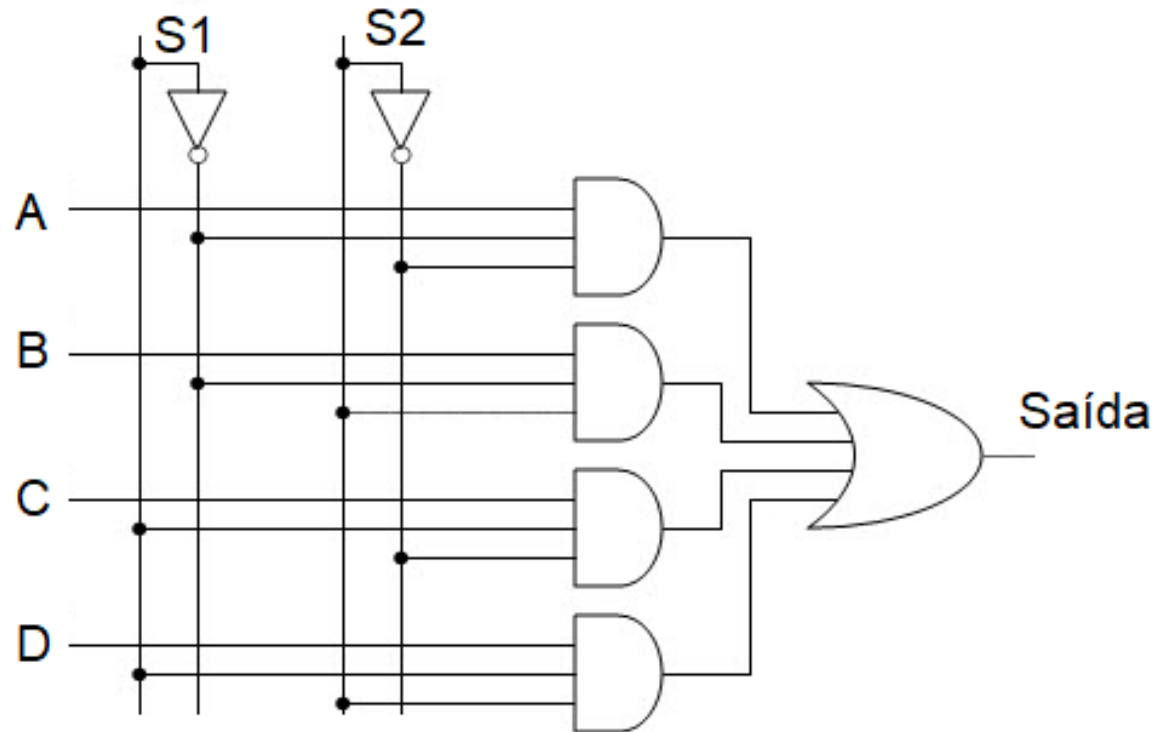
ENTITY Primeiro_projeto IS
  PORT (IN1 : IN std_logic;
        IN2 : IN std_logic;
        IN3 : IN std_logic;
        SAIDA1 : OUT std_logic;
        SAIDA2 : OUT std_logic);
END Primeiro_projeto;

ARCHITECTURE logica OF Primeiro_projeto IS
BEGIN
  SAIDA1<= (IN1 AND IN2) OR IN3;
  SAIDA2<= (IN1 XOR IN2) AND NOT (IN3);
END logica;
```



VHDL – Exercício 2

- Escreva o código e faça a simulação do multiplexador 4:1 abaixo:



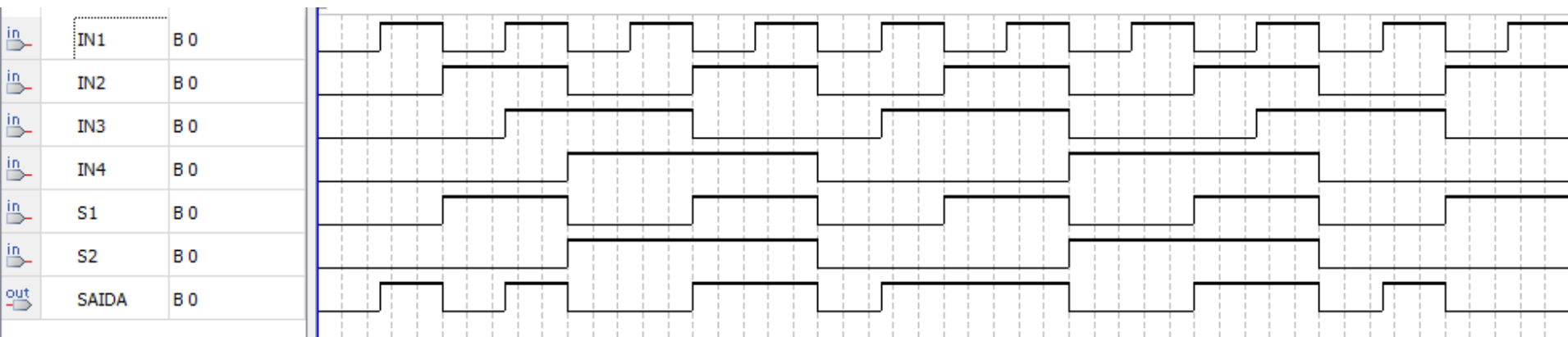
VHDL – Exercício 2

▪ Resposta:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

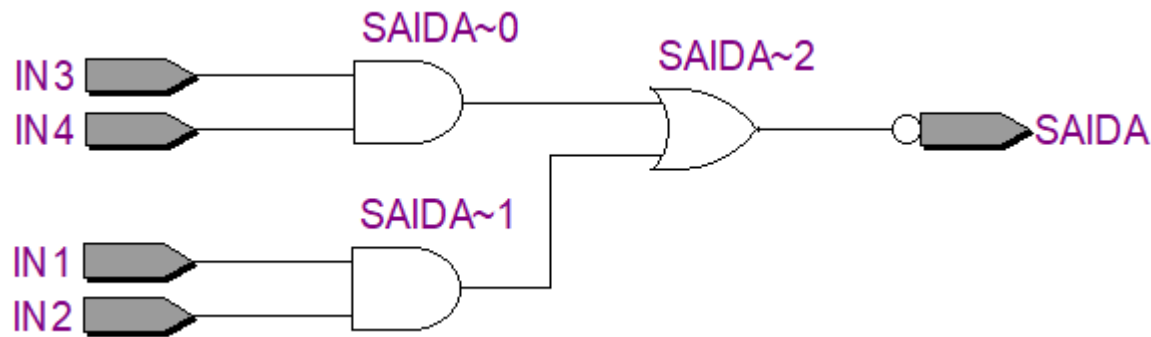
ENTITY mux4_1 IS
    PORT (IN1, IN2, IN3, IN4, S1, S2 : IN std_logic;
          SAIDA : OUT std_logic);
END mux4_1;

ARCHITECTURE logica OF mux4_1 IS
BEGIN
    SAIDA <= ((IN1 and not(S1) and not(S2)) OR
              (IN2 and not(S1) and S2) OR (IN3 and S1 and not(S2)) OR
              (IN4 and S1 and S2));
END logica;
```



VHDL – Exercício 3

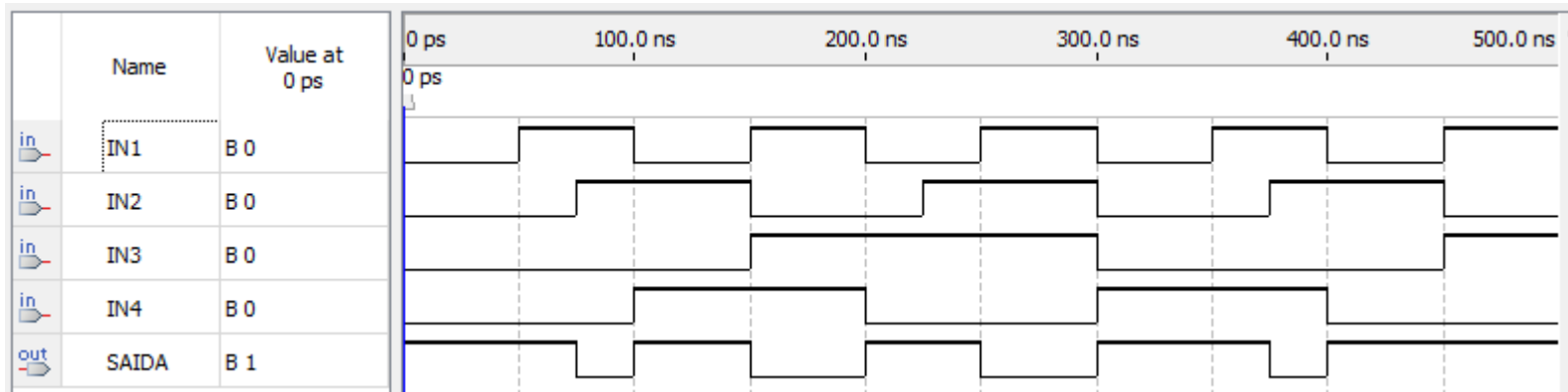
- Escreva o código e faça a simulação do circuito combinacional abaixo:



VHDL – Exercício 3

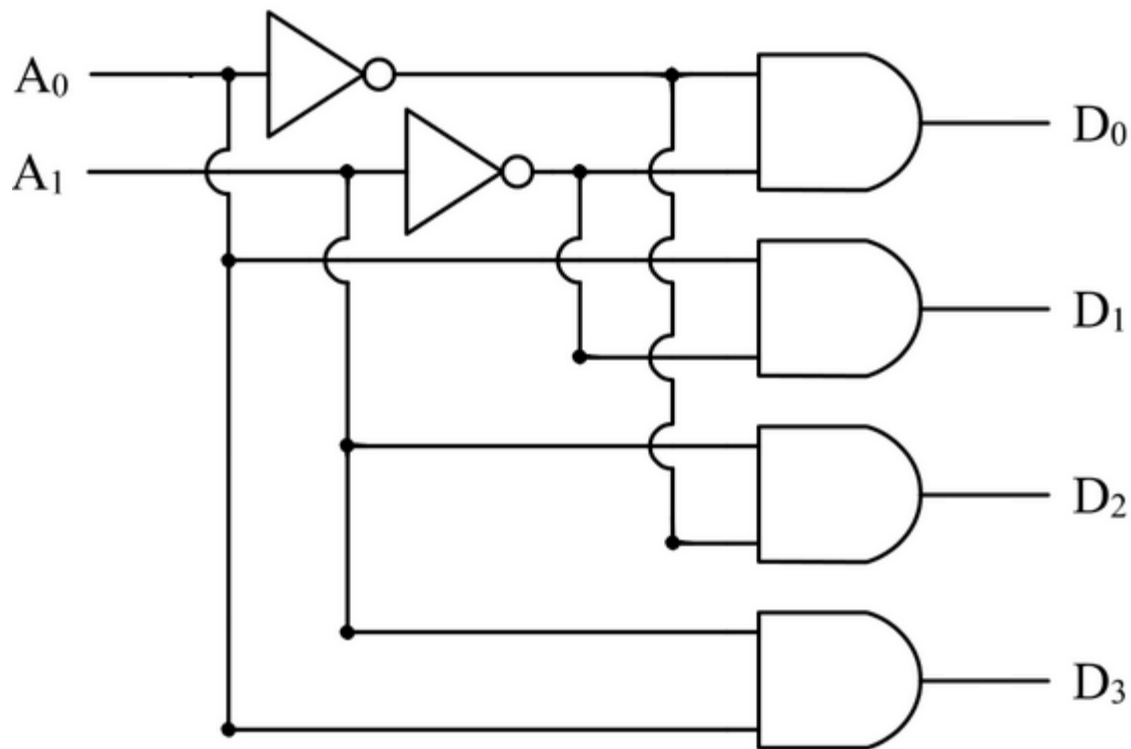
▪ Resposta:

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY circ_comb IS  
    PORT (IN1, IN2, IN3, IN4 : IN std_logic;  
          SAIDA : OUT std_logic);  
END circ_comb;  
  
ARCHITECTURE logica OF circ_comb IS  
BEGIN  
    SAIDA <= NOT((IN3 AND IN4) OR (IN1 AND IN2));  
END logica;
```



VHDL – Exercício 4

- Faça o decodificador 2:4 abaixo

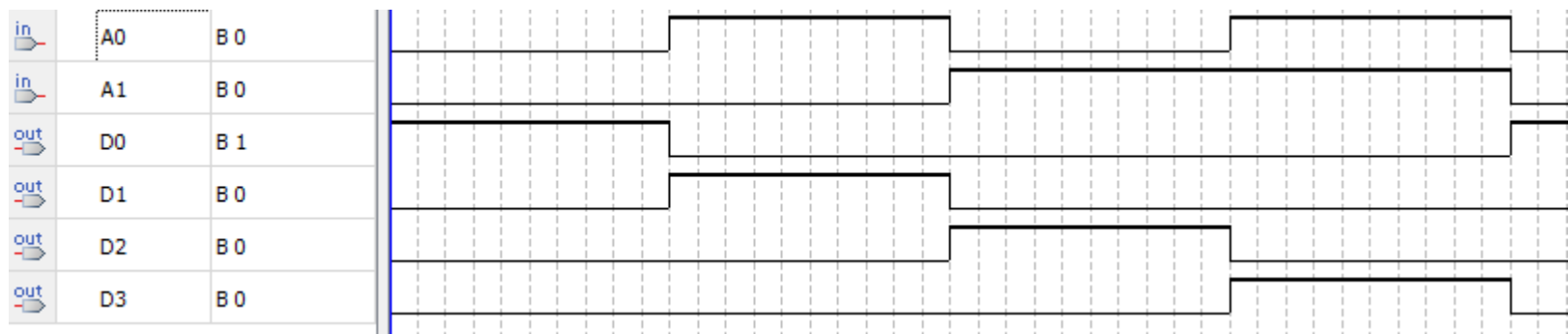


A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

VHDL – Exercício 4

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY ex1 IS  
    PORT (A0, A1 : IN std_logic;  
          D0, D1, D2, D3 : OUT std_logic);  
END ex1;  
  
ARCHITECTURE logica OF ex1 IS  
BEGIN  
    D0<=NOT (A0) AND NOT (A1);  
    D1<=A0 AND NOT (A1);  
    D2<=NOT (A0) AND A1;  
    D3<=A0 AND A1;  
END logica;
```

A ₁	A ₀	D ₃	D ₂	D ₁	D ₀
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



- Próxima aula: Utilizando o kit de desenvolvimento.