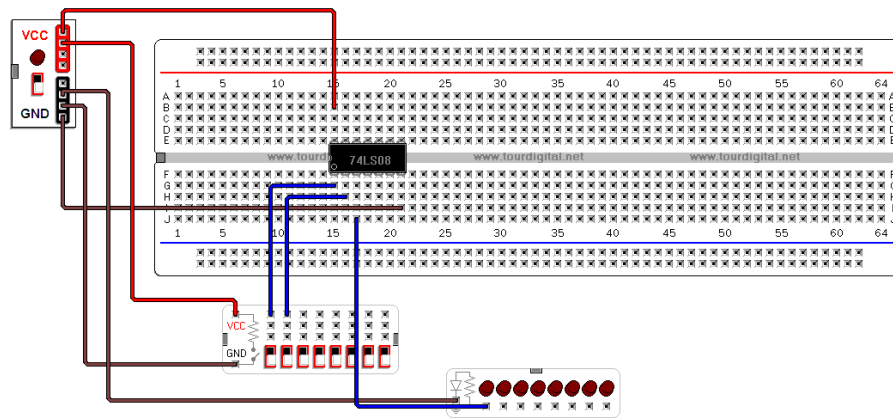


Conteúdo Prático - PED



Aluno: João Eduardo Pereira Rabelo

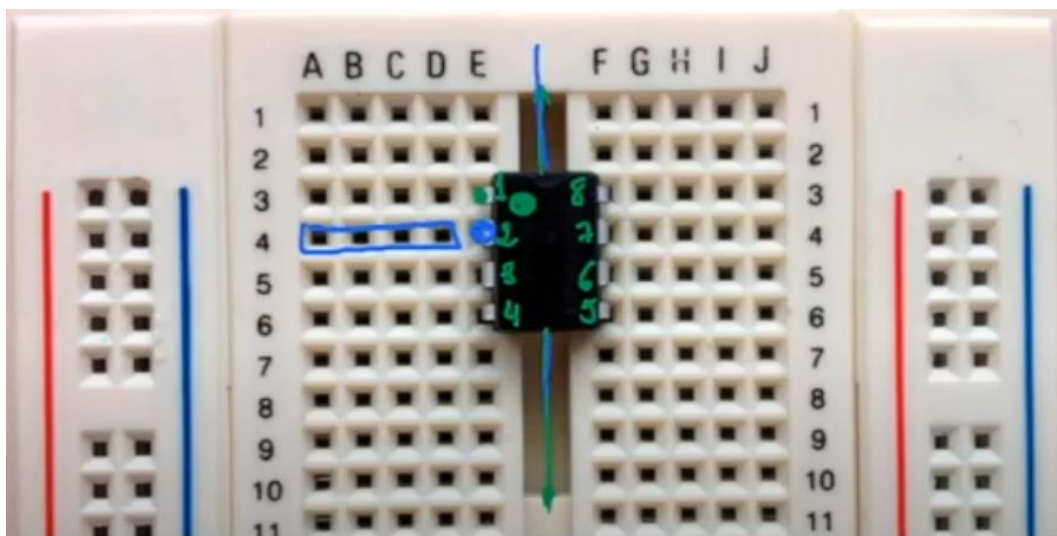
Matrícula: 180053299

- ☐ Módulos da Disciplina
- ☒ 1 - Introdução à disciplina
- ☒ 2 - Combinacionais - 1
- ☒ 3 - Combinacionais - 2
- ☒ 4 - Combinacionais - 3
- ☒ 5 - Combinacionais - 4
- ☐ 6 - Combinacionais - 5
- ☐ 7 - Combinacionais - 6
- ☐ 8 - Aritméticos - 1
- ☐ 9 - Aritméticos - 2
- ☐ 10 - Sequenciais - 1
- ☐ 11 - Sequenciais - 2
- ☐ 12 - Sequenciais - 3
- ☐ 13 - Sequenciais - 4

ProtoBoard



- Barramentos: Parte Superior e Inferior da Protoboard que é responsável pela passagem de energia (polo + para o polo -).
 - Não são interligadas, geralmente, necessitando 2 “jumpers” para serem usados em conjunto, um no positivo de um lado e no positivo de outro e outro no negativo de um lado no negativo do outro.
 - No meio de cada barramento, geralmente são separados nas protoboards, necessitando “jumpers” para ligar um lado ao outro.
- CI's são geralmente acoplados no vão do meio da placa
- A numeração dos slots da protoboard, seguem o seguinte padrão:



- Sendo cada coluna e linha separada das do lado oposto, mas entre si interligadas em suas respectivas linhas (Ex: A1, B1, C1, D1 e E1).

Multímetro



- Medidor de tensão.
 - Utilizado para medir a tensão (V) correndo pelo sistema.



- Amperímetro
 - Apenas para medir corrente (A), NÃO usar com tensão correndo pelo sistema, pois irá queimar a ferramenta.



- Ohmímetro
 - Apenas para medir a resistência (Ω), NÃO deve ser usado com corrente passando no circuito.



- Teste de Continuidade
 - Para testar LED's ou outro circuito, serve para verificar se tem continuidade entre os dois terminais do multímetro.



Resistor



- Calcular a medida do Resistor
 - Verificar a tensão da corrente e dos componentes do circuito.
 - $V_r = R * I$
 - V_r : Tensão da Corrente - Tensão do Circuito.
 - I : Corrente em Miliamperes.
 - R : Resistor
- Leitura do Resistor

4- Faixas

1.0 K Ω \pm 5%

1st 2nd 3rd 4th

Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador Decimal		Tolerância
Preto	0	0	0	1	1	
Marrom	1	1	1	10	10	\pm 1%
Vermelho	2	2	2	100	100	\pm 2%
Laranja	3	3	3	1K	1.000	
Amarelo	4	4	4	10K	10.000	
Verde	5	5	5	100K	100.000	
Azul	6	6	6	1M	1.000.000	
Violeta	7	7	7	10M	10.000.000	
Cinza	8	8	8		100.000.000	
Branco	9	9	9		1.000.000.000	
Ouro				0.1		\pm 5%
Prata				0.01		\pm 10%
Branco						\pm 20%

1st 2nd 3rd 4th 5th

254 Ω \pm 1 %

5- Faixas

- $(1^{\text{a}} \text{ Faixa} + 2^{\text{a}} \text{ Faixa}) * 3^{\text{a}} \text{ Faixa}$ – Com tolerância da 4ª Faixa.

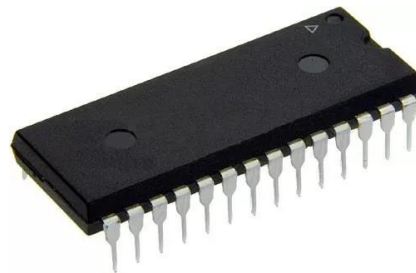
Obs: (Não é somatória (Ex: 1+5=15))

LED

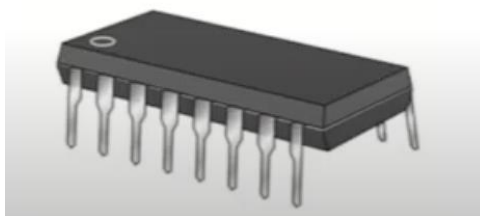


- Sentido Padrão: + / - (ou caso queira inverter a tabela verdade - / +)
- Tensão (Máxima):
 - Vermelho, Amarelo e Verde: 2V
 - Azul e Branco: 3V
 - Se a tensão foi maior que 5V é NECESSÁRIO um resistor
- Terminal Positivo -> Chanfro ou a perna mais longa.
- Terminal Negativo -> Circunferência normal ou a perna mais curta.

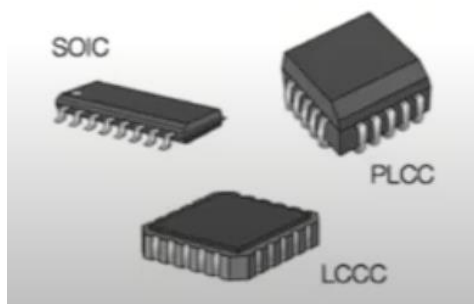
Circuitos Integrados



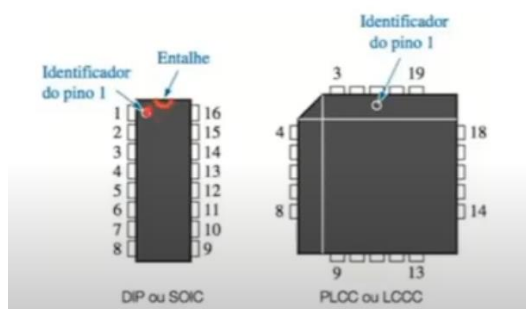
- DIP
 - Os pinos são inseridos em furos que podem ser soldados na trilha do lado oposto da placa



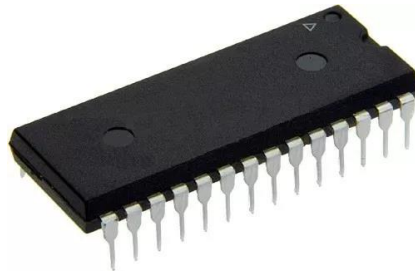
- SMT
 - Não precisa de furos na placa, os pinos são soldados diretamente na trilha das placas



- Pinagem
 - Contagem a partir do entalhe do circuito, no sentido ante horário

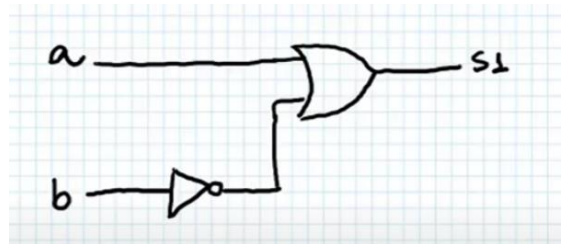


Subfamília TTL e CMOS



- Não se pode misturar circuitos TTL e CMOS devido a diferença de tensão para definir estado “0” e “1”.
 - TTL
 - Alto – entre 5V e 2,4V
 - Proibida - entre 2,4V e 0,5V
 - Baixo – entre 0,5V e 0,3V
 - CMOS
 - Alto – entre 13,5V e 12,5V
 - Proibida - entre 12,5V e 2,5V
 - Baixo – entre 2,5V e 1,5V

Portas Lógicas



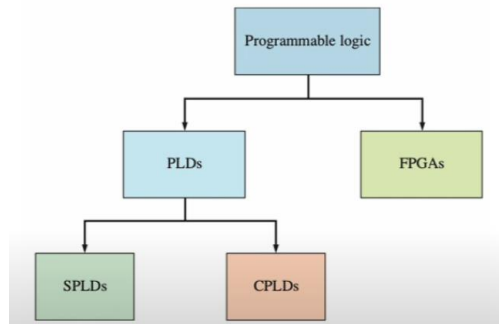
- Quando tratamos um circuito com portas lógicas, podem ser demonstrados em desenho, conforme acima, ou escrita para utilização da tabela verdade.

○ Ex: $S = a + \bar{b}$

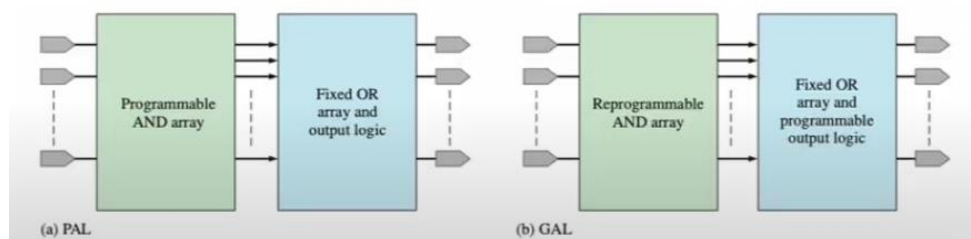
- Ou seja, a saída do circuito seria positiva caso exista, uma entrada positiva “a” ou uma entrada não positiva “b”.
- Conforme a Tabela Verdade:

A	B	S
V	V	V
V	F	V
F	V	F
F	F	V

Lógica Programável



- Vimos circuitos de lógica simples anteriormente (Ex: Circuito AND TTL) mas nesse módulo, iremos abordar o uso e funcionamento de circuitos de lógica programável.
- Tipos:
 - PLDs – Dispositivo de Lógica Programável.
 - SPLDs – Dispositivo de Lógica Programável Simples.



SPLDs com uma matriz de portas AND e OR do tipo PAL e GAL.

- PAL – Array Lógico Programável
 - Pode ser programada uma única vez.
- GAL – Array Lógico Genérico
 - Pode ser reprogramado várias vezes.

- CPLDs – Dispositivo de Lógica Programável Complexo.

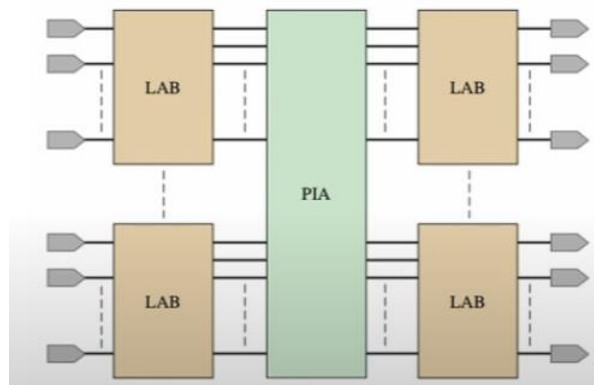
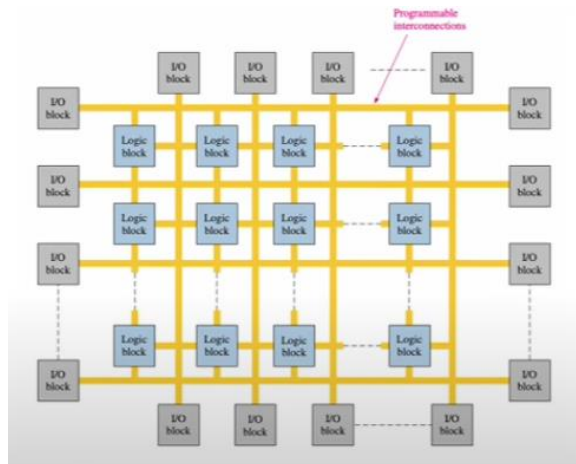


Diagrama genérico de um CPLD

- Múltiplos SPLDs interligados entre si.
- LAB = SPLDs
- PIA = Matriz Programável de Interconexão.

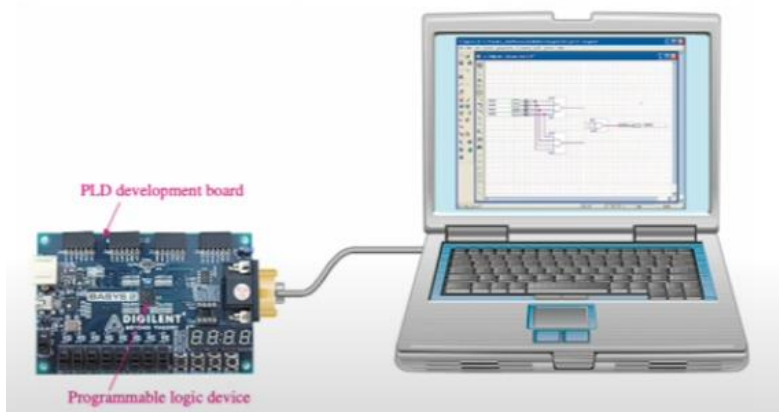
- FPGAs – Campo Programável de Portões de Array



Estrutura básica de uma FPGA

- Blocos lógicos interligados entre si e aos blocos de saída através de uma interconexão programada e classificados de acordo com a granularidade:

- Grão Grosso
 - Blocos Lógicos mais complexos e em menor quantidade.
- Grão Fino
 - Blocos lógicos de complexidade relativamente mais simples, mas em quantidade maior
- Programação dos Circuitos de Lógica Programável



Exemplo de um setup para programação.

- Feita através de um software específico fornecido pelo fabricante.
 - Software usado vai ser o VIVADO.
- Fluxo de Projeto
 - Descrição HDL
 - Especificação
 - Definição do problema.
 - Através dessa especificação é feito uma descrição HDL do circuito.
 - Simulação Funcional
 - Verificar se aquela descrição tem correspondência com a especificação.
 - Síntese
 - Gera um circuito que atenda a descrição, resultando em uma NetList.

- Implementação
 - Gera um novo circuito a partir da netlist da etapa anterior.
 - Place and Route
 - Arquivo gerado com uma temporização dos componentes
- Simulação com Atraso
 - Nova simulação levando em consideração com os mesmos estímulos, mas com os atrasos de propagação definidos no Place and Route.
- Programação
 - Gerado um arquivo blitstream que é transmitido via USB para a FPGA.

VHDL

HDL – Hardware Description Language

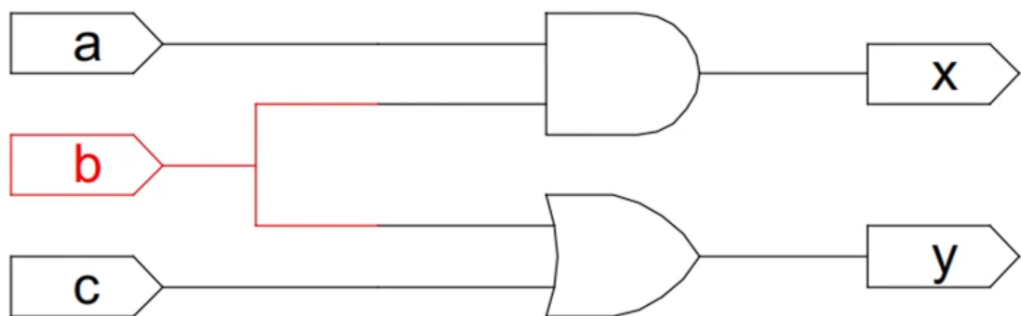


```
entity and_gate is
port (
a, b : in bit;
z : out bit
);
end and_gate ;

architecture logic of and_gate is
begin
z <= a and b;

end architecture ; -- logic
```

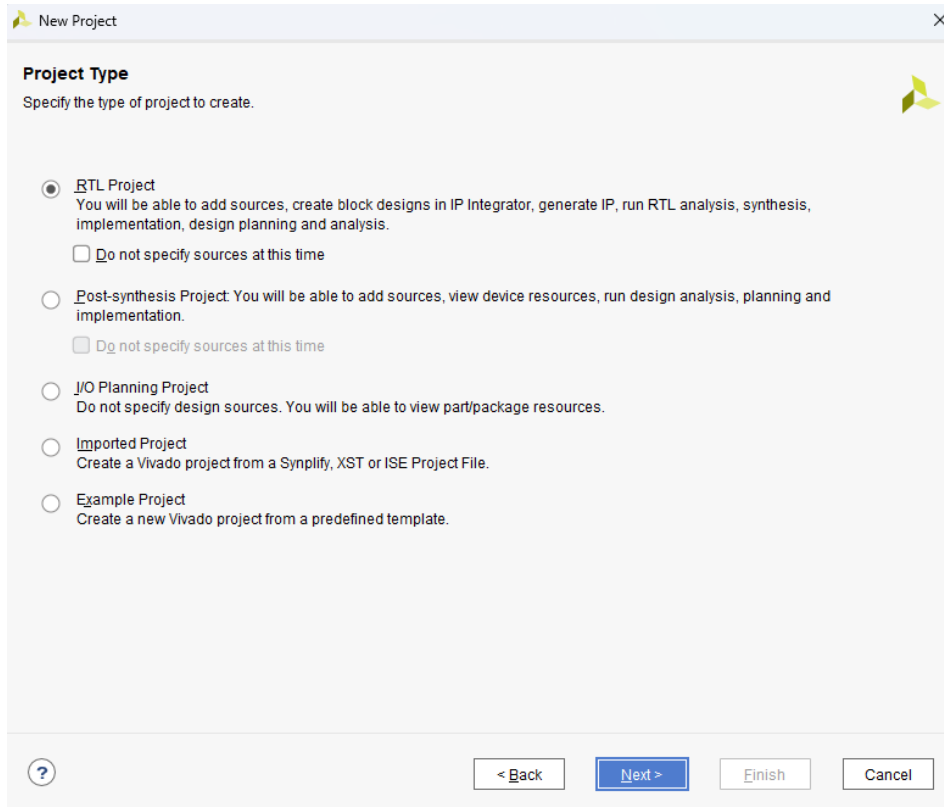
- Exemplo de sintaxe de um circuito usando VHDL



```
1  ENTITY portas IS
2      PORT (a, b, c : IN  BIT;
3              x, y   : OUT BIT);
4  END portas;
5
6  ARCHITECTURE teste OF portas IS
7  BEGIN
8      x <= a AND b;
9      y <= c OR  b;
10 END teste;
```

- Usando o VIVADO

- Com o VIVADO aberto, será necessário criar o projeto.
- Selecione o Tipo de Projeto (Exemplo abaixo é o padrão)



New Project

Project Type
Specify the type of project to create.

☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

☐ **Post-synthesis Project**: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

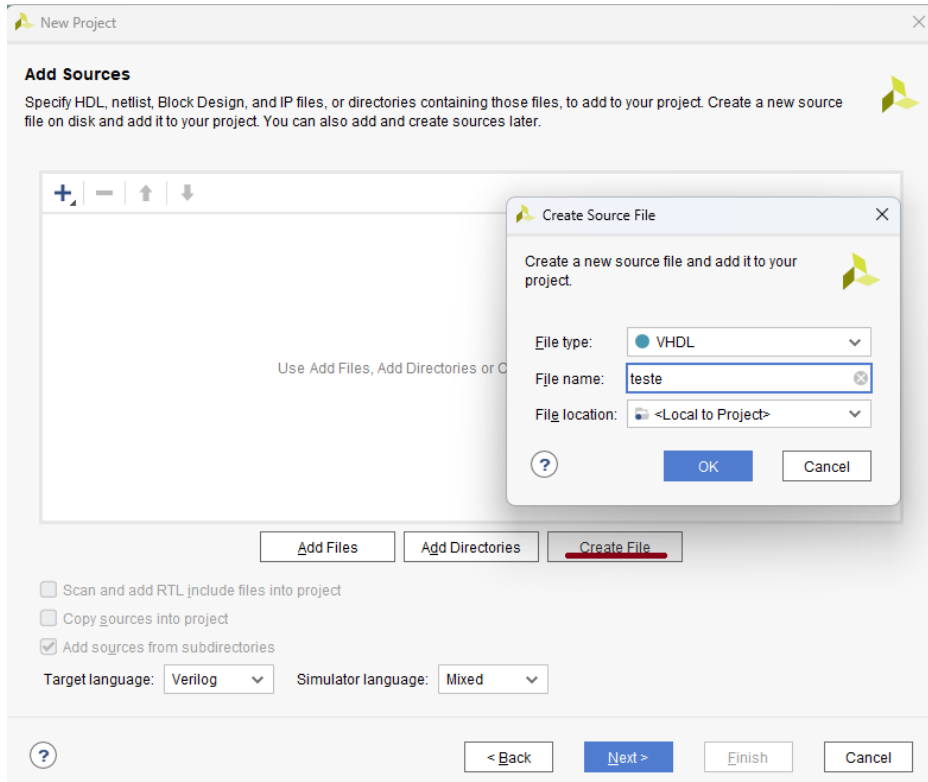
☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

- Crie o(s) arquivo(s) de design do projeto.



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Use Add Files, Add Directories or Create Source File

Create Source File

Create a new source file and add it to your project.

File type: **VHDL**

File name: **teste**

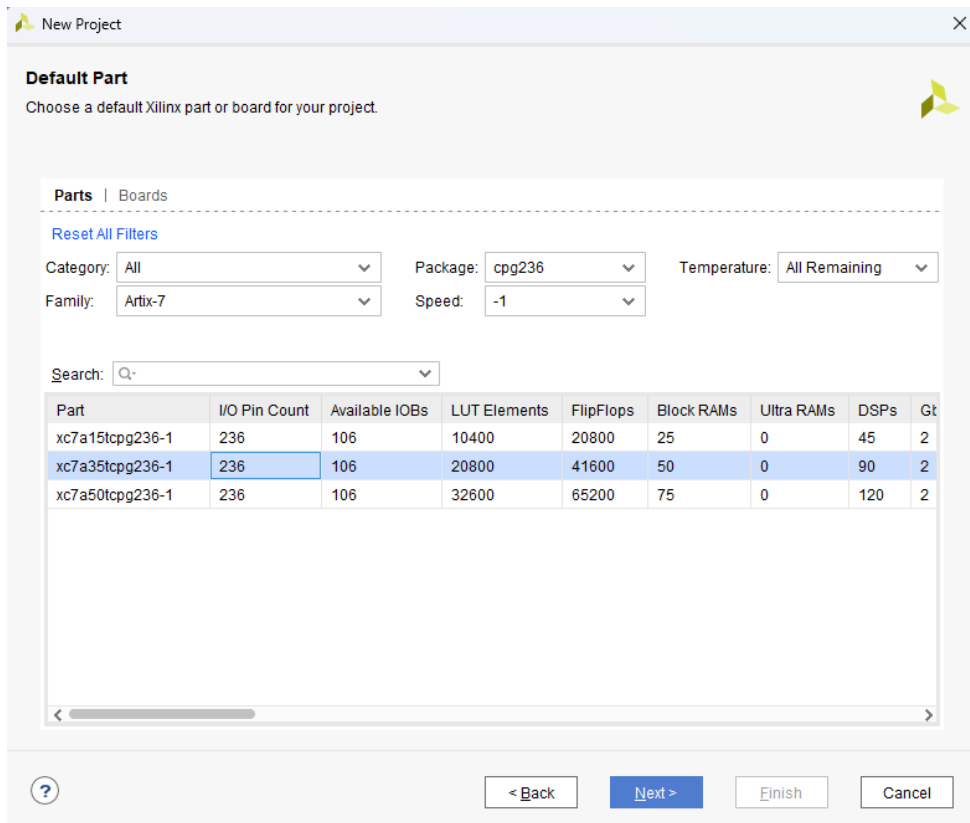
File location: **<Local to Project>**

OK Cancel

☐ Scan and add RTL include files into project
☐ Copy sources into project
☒ Add sources from subdirectories

Target language: **Verilog** Simulator language: **Mixed**

- Selecione a Placa Basys do Projeto.



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

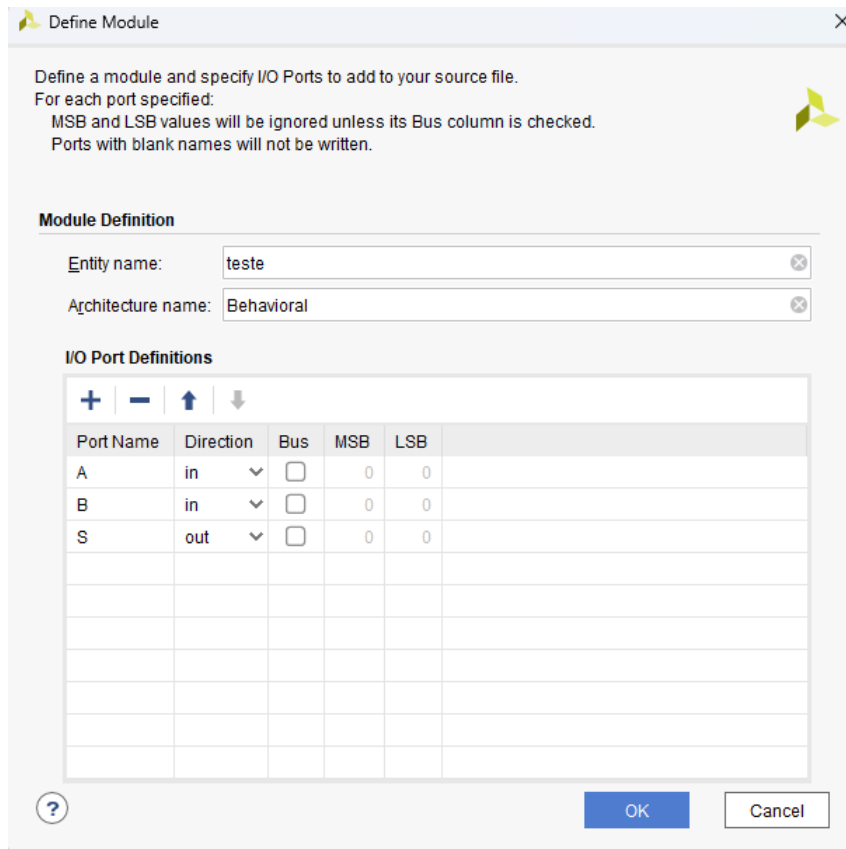
Category: **All** Package: **cpg236** Temperature: **All Remaining**

Family: **Artix-7** Speed: **-1**

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7a15tcpg236-1	236	106	10400	20800	25	0	45	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2
xc7a50tcpg236-1	236	106	32600	65200	75	0	120	2

- Insira os módulos de entrada e saída do projeto.



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name: teste

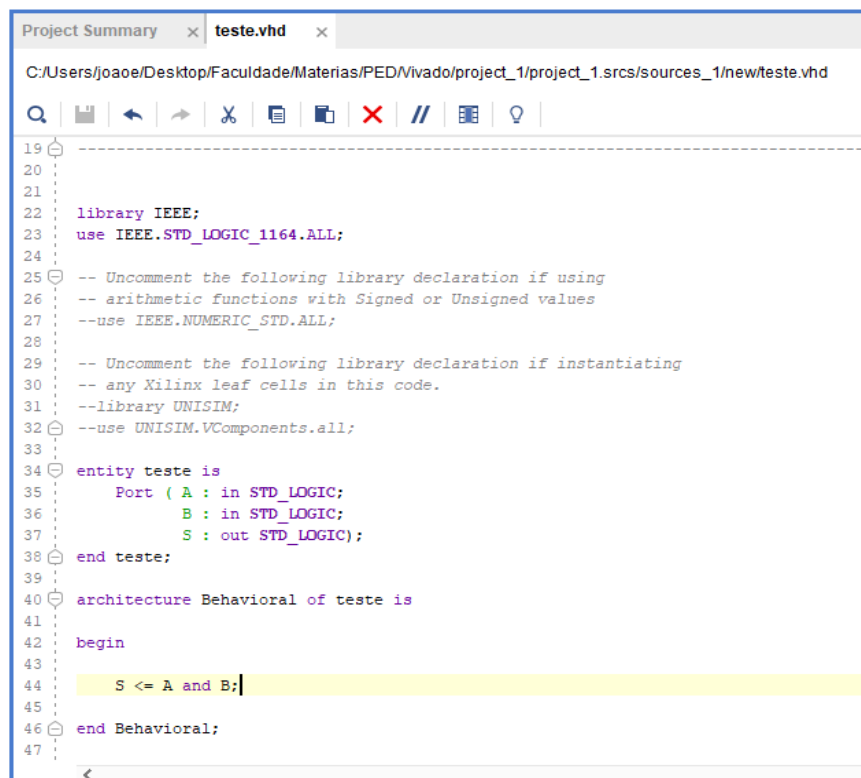
Architecture name: Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
A	in	<input type="checkbox"/>	0	0
B	in	<input type="checkbox"/>	0	0
S	out	<input type="checkbox"/>	0	0

OK Cancel

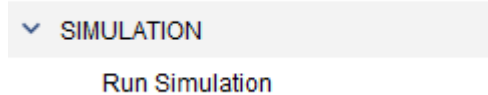
- Selecione o arquivo Design que foi criado e trabalhe com ele.



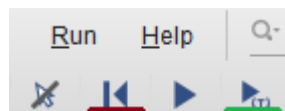
```

19 -----
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity teste is
35     Port ( A : in STD_LOGIC;
36           B : in STD_LOGIC;
37           S : out STD_LOGIC);
38 end teste;
39
40 architecture Behavioral of teste is
41
42 begin
43
44     S <= A and B;
45
46 end Behavioral;
47
  
```

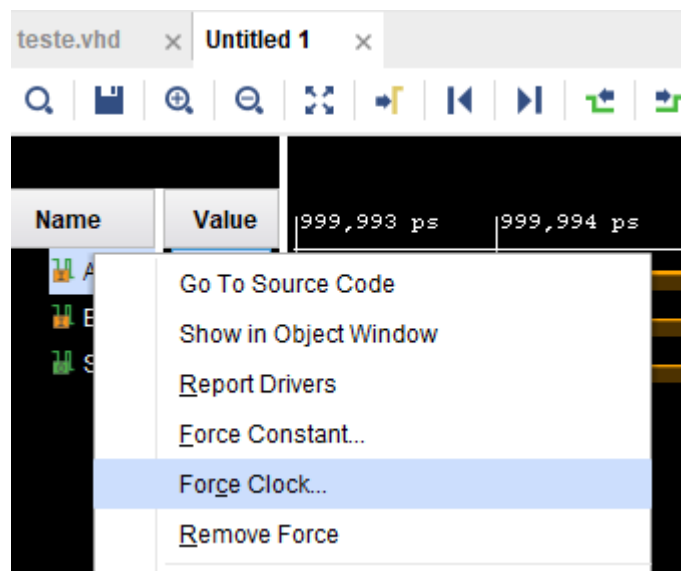
- Para simular o funcionamento do circuito clique em “Run Simulation” no canto esquerdo do programa.



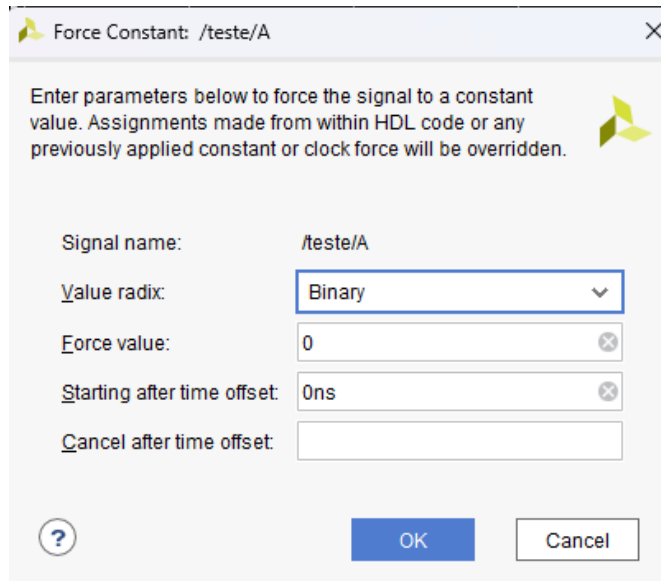
- Antes de dar continuidade, confirme se o arquivo .vhd está salvo e resete a simulação (Vermelho).



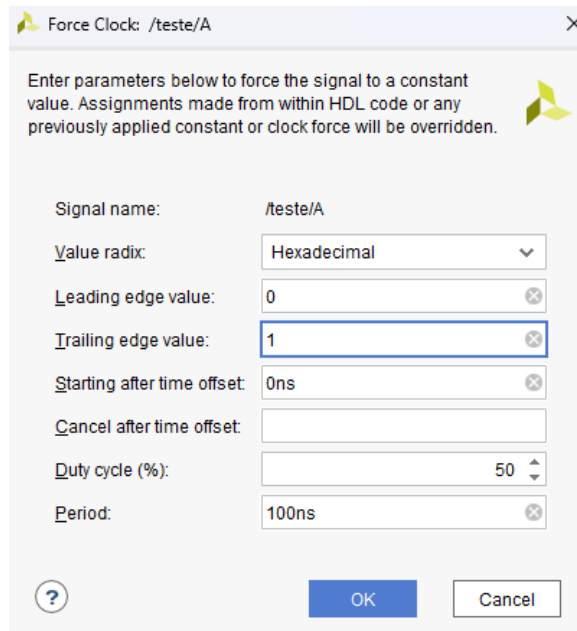
- Na Simulação será necessário clicar nas entradas com o botão direito e dar um “Force Constant” ou “Force Clock”



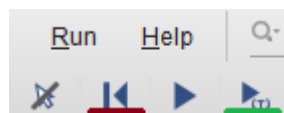
- Para definir o valor esperado no “Force Constant” siga o passo abaixo:



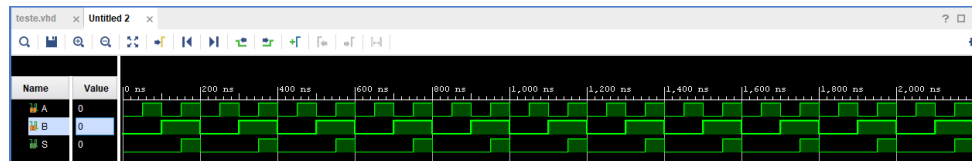
- Para Definir o valor esperado no “Force Clock” siga o passo abaixo:



- Feito as configurações, rode o programa (Verde).

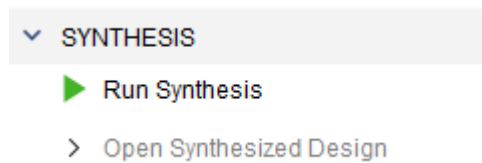


- Caso tenha feito o “Force Clock”, deverá aparecer a seguinte análise.

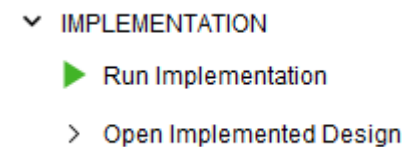


- Para executar o código desenvolvido no VIVADO na placa BASYS, gerando o arquivo Bitstream (conforme citado nos tópicos acima) vai ser preciso o seguinte passo a passo:

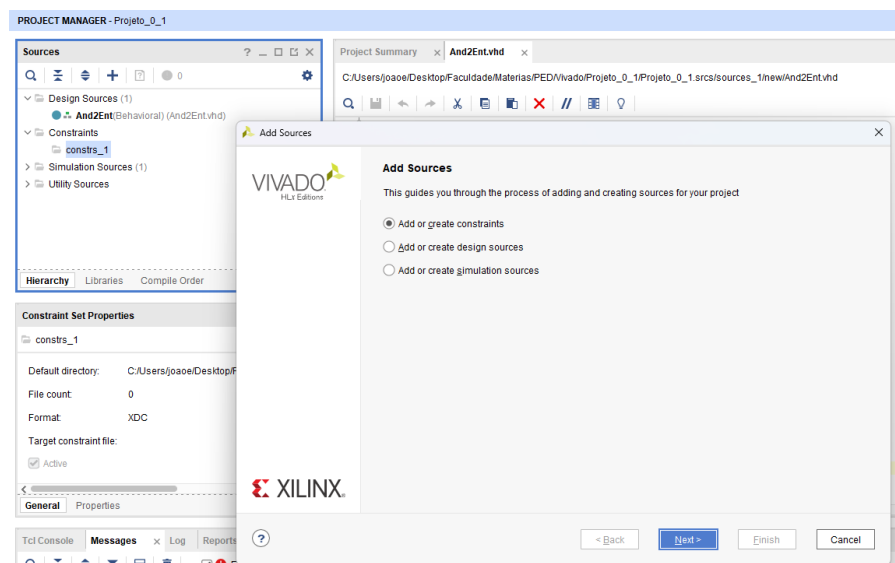
- Rodar uma Síntese do código



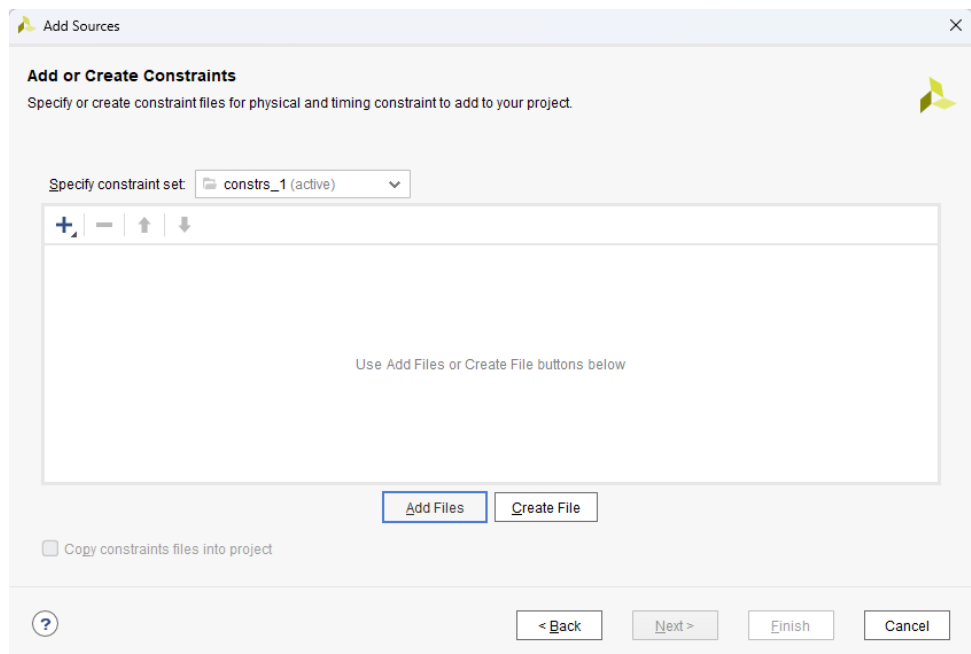
- Rodar uma Implementação do Código



- Antes de gerar o Bitstream, deverá ser gerado o arquivo de Constraints que vai ser o link do HDL e a Placa.



- Irá ser adicionado o arquivo .xdc da Placa (no caso de PED, está no Aprender), clicando em “Add Files” e marque a opção “Copy Constraints files into project”.



- Adicionando o arquivo e abrindo, será necessário editar para o uso no projeto, no caso abaixo, as entradas “A” e “B”, serão usadas pelos Switches (No caso o V16 e V17, mas podem ser outros, decididos pelo aluno).

```

11 | ## Switches
12 | #set_property PACKAGE_PIN V17 [get_ports {sw[0]]}
13 |     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]]}
14 | #set_property PACKAGE_PIN V16 [get_ports {sw[1]]}
15 |     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]]}

```

Ficando assim:

```

12 | set_property PACKAGE_PIN V17 [get_ports {A}]
13 |     set_property IOSTANDARD LVCMOS33 [get_ports {A}]
14 | set_property PACKAGE_PIN V16 [get_ports {B}]
15 |     set_property IOSTANDARD LVCMOS33 [get_ports {B}]

```

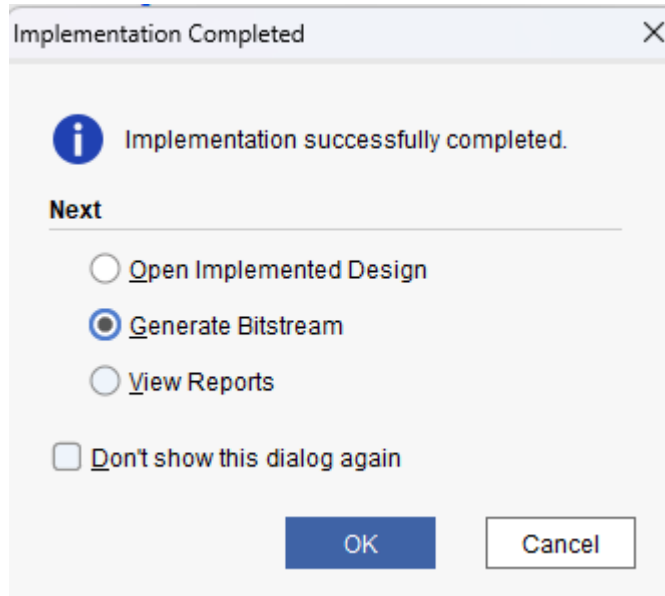
- A saída “S” será nos LEDS da placa que ficam em frente aos Switchs, tendo que ser verificado e alterado também, conforme exemplo:

```

47 | set_property PACKAGE_PIN U16 [get_ports {S}]
48 |     set_property IOSTANDARD LVCMOS33 [get_ports {S}]

```

- Dando tudo certo, pode ser mandado gerar o Bitstream



- Gerado o Bitstream, verifique se na placa BASYS 3, estes pinos estão nas seguintes configurações:

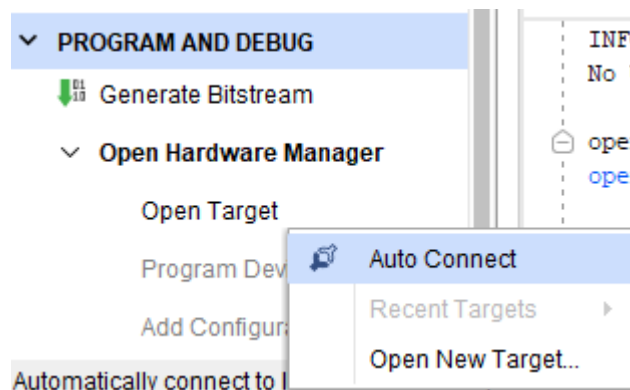


Jumper precisa estar no “USB”

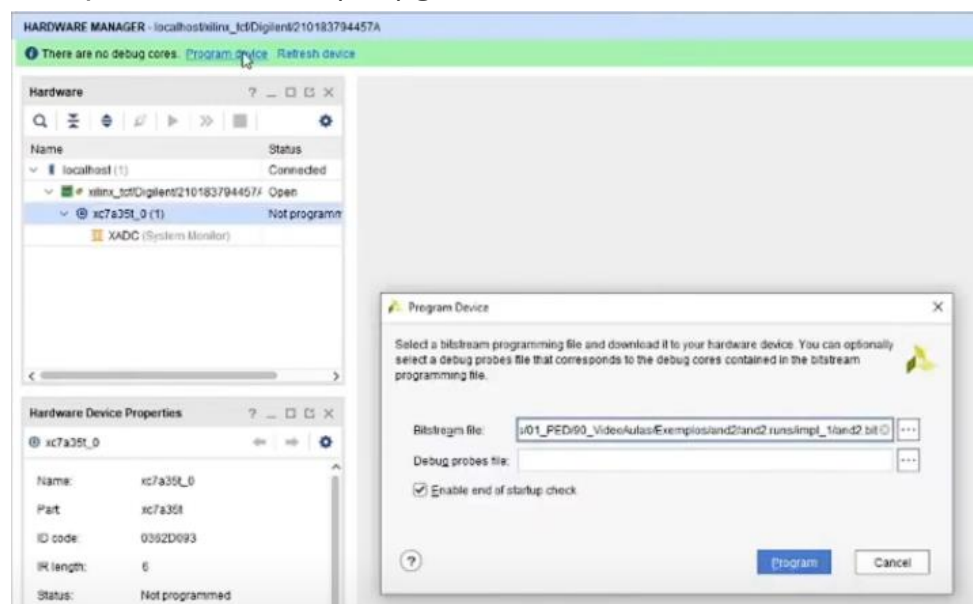


Jumper precisar estar no “JTAG”

- Feito tudo isso, será necessário conectar com a placa (no Windows pode dar erro por questão de Drivers, verificar vídeo no aprender ou consultar professor)



- Feito a conexão, será necessário programar a placa com o arquivo Bitstream (.bit) gerado



- Feito isso, a placa está programada e pronta para demonstração/uso.