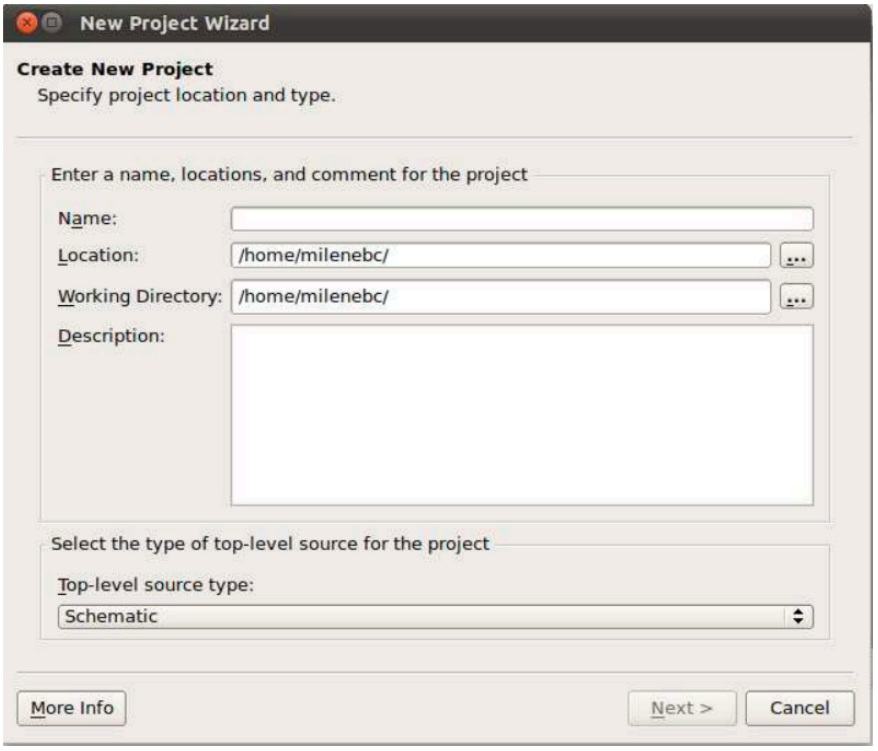
Guia FPGA - VHDL

O objetivo deste documento é servir como um guia rápido para o projeto de circuitos em FPGA utilizando VHDL. O guia pode ter seu conteúdo alterado a cada novo conceito apresentado na disciplina de Laboratório de Sistemas Digitais e Computacionais. Esse documento foi produzido e compartilhado no Google Drive para que todos os alunos possam editá-lo colaborando para o desenvolvimento de um guia que possa auxiliar a todos na utilização do FPGA no Laboratório de Hardware da UFSJ e no desenvolvimento de códigos em VHDL.

FPGA e ISE

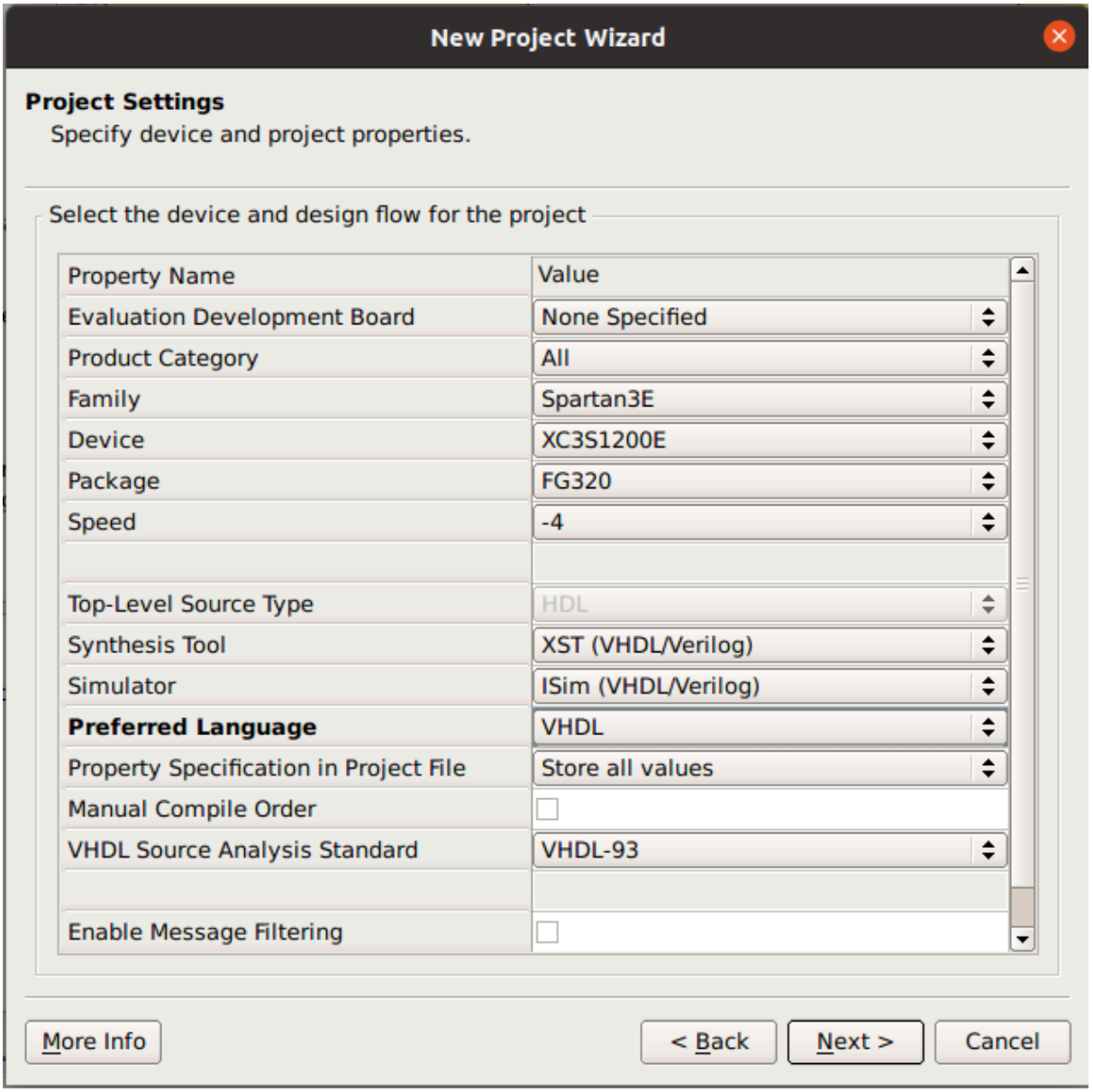
Criar um projeto no ISE

− Acesse o menu File → New Project



− Dê um nome para o projeto no campo *Name* (o ISE criará uma pasta com esse nome e todos os arquivos do projeto serão armazenados nessa pasta).
IMPORTANTE: nunca utilize espaços, símbolos e caracteres especiais para dar nomes a arquivos e pastas no ISE. Para separar nomes, utilize o *underline* (_)
− Selecione a pasta em que o projeto estará localizado no campo *Location* (é necessário ter permissão de escrita na pasta e não deve existir pasta com espaços ou caracteres especiais no caminho). Se necessário, o botão (...) pode ser utilizado para modificar a pasta destino. Recomenda-se que o campo *Working Diretory* seja o mesmo do campo *Location*.

− Na caixa *Top level source type* selecione HDL (para um projeto de circuito usando linguagem de descrição de hardware) ou *Schematic* (para um projeto de circuito usando esquemático).
− Clique no botão Next>.
− Na caixa de configuração do dispositivo e do projeto, modifique as configurações para que sejam exatamente iguais às apresentadas na figura a seguir e pressione o botão Next>.



Criar módulo (circuito ou subcircuito) VHDL no ISE

− Clique com o botão direito do *mouse* sobre qualquer área do painel *Hierarchy*.
− Selecione *New source* no menu que aparecer.
− Na caixa de diálogo *New Source Wizard* selecione a opção *VHDL Module*.
− Dê um nome (sem espaços, símbolos e caracteres especiais) para o módulo na caixa *File name.*
− Verifique se a caixa Location contém a pasta do seu projeto.

− Verifique se o *checkbox Add to project* está marcado.
− Clique em *Next>.*
− Será apresentado um resumo do projeto. Clique no botão *Finish*.

Criar *testbench*

− Clique com o botão direito do *mouse* sobre qualquer área do painel *Hierarchy*.
− Selecione *New source* no menu que aparecer.
− Na caixa de diálogo *New Source Wizard* selecione a opção *VHDL Test Bench*.
− Dê um nome (sem espaços, símbolos e caracteres especiais) para o módulo na caixa *File name*. Sugere-se que o nome seja o mesmo do circuito seguido por tb. Por exemplo, circuito_tb.
− Verifique se a caixa Location contém a pasta do seu projeto.
− Verifique se o *checkbox Add to project* está marcado.
− Clique em *Next>.*
− Selecione o módulo que você deseja testar.
− Caso o circuito a ser testado **não** possua sinal de clock dentre suas entradas, remova as linhas a seguir:

- 57-60:

```
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant <clock>_period : time := 10 ns;
```

- 71-78:

```
-- Clock process definitions
<clock>_process :process
begin
        <clock> <= '0';
        wait for <clock>_period/2;
        <clock> <= '1';
        wait for <clock>_period/2;
end process;
```

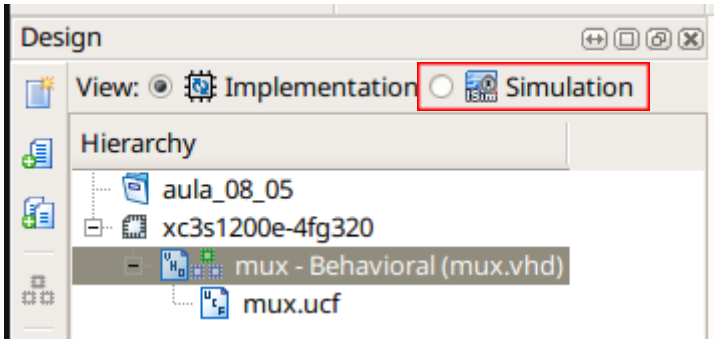
- 87:

```
    wait for <clock>_period*10;
```

− Insira o código de estímulos (valores atribuídos às entradas do circuito) abaixo da linha 89:

```
    -- insert stimulus here
```

− Mude a visualização para *Simulation* na opção *View*, sobre o painel *Hierarchy*, clicando no campo destacado na figura a seguir.



− Selecione o módulo *testbench* (circuito_tb) no painel *Hierarchy*.
− No painel *Processes* expanda a opção *ISim Simulator*, clicando no símbolo +, e clique 2 vezes em *Simulate Behavioral Model*.

− Na janela *ISim*, ajuste o zoom da simulação clicando em  e, se necessário, selecione uma nova área de zoom e clique em  .
− Se necessário, ajuste o padrão de exibição de valores de um sinal clicando com o botão direito do mouse sobre o nome do sinal, depois clique em *Radix* e selecione o formato desejado.

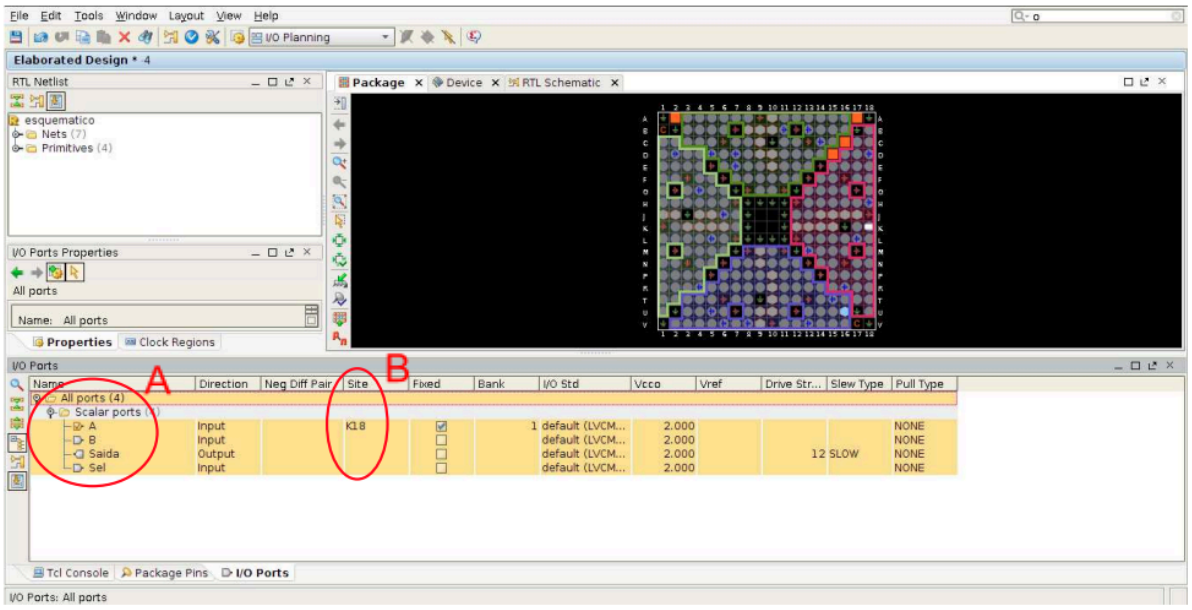
Caso após a geração do *testbench* você altere as entradas ou saídas do circuito testado, você deverá modificar o componente, a declaração de sinais e o `port map` no *testbench* para refletir as alterações realizadas.

Trocar o módulo principal do projeto

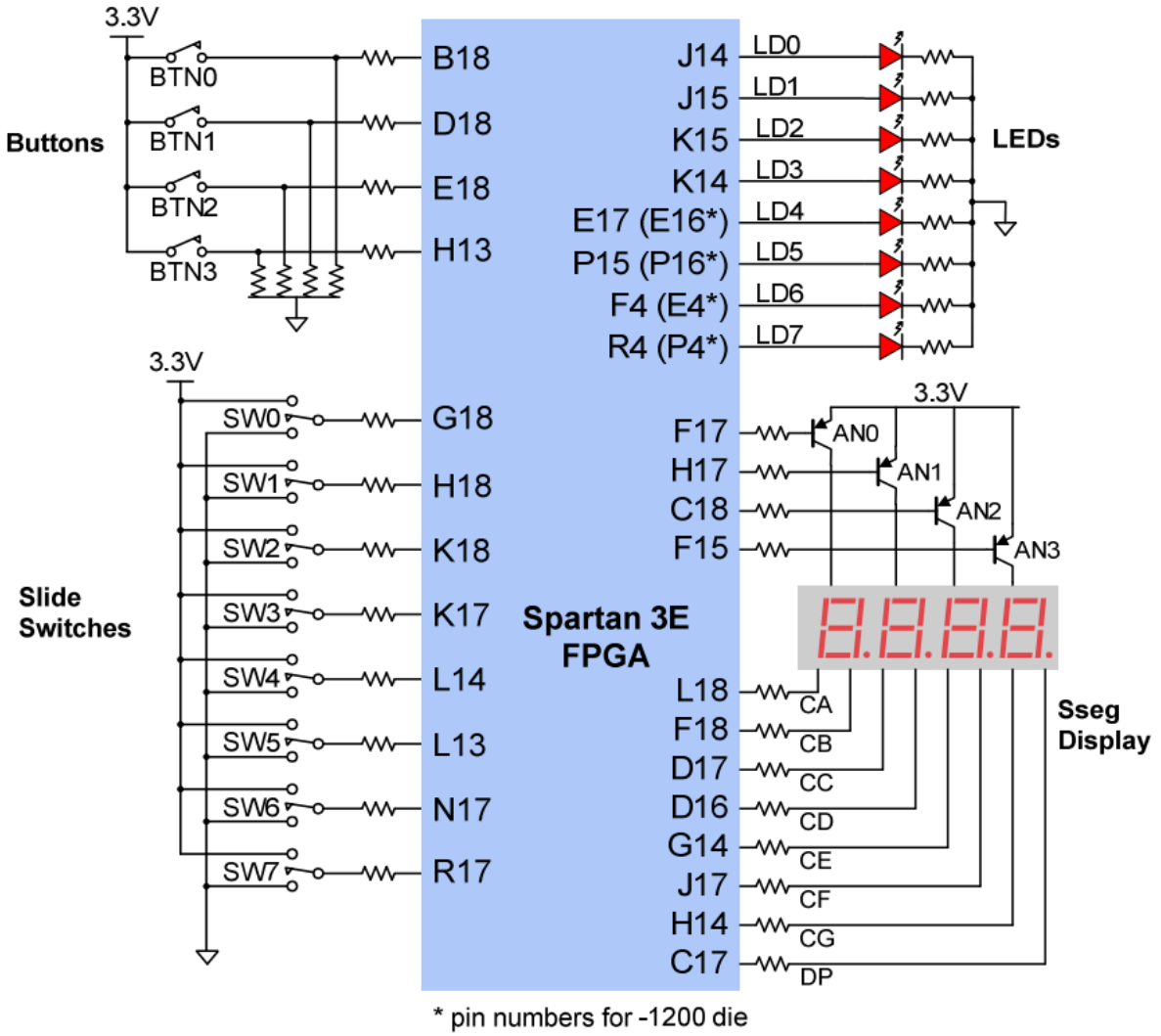
O módulo principal do projeto e para o qual será gerado o *bitstream* tem ao lado de seu nome, no painel *Hierarchy* o símbolo  . Para trocá-lo, clique com o botão direito sobre o módulo desejado e selecione a opção *Set as Top Module*.

Associar os pinos do FPGA às entradas e saídas

− Verifique se o módulo principal do projeto está correto (no painel *Hierarchy* o símbolo  fica a frente deste módulo). Caso esteja incorreto, faça o procedimento da seção “Trocar o módulo principal do projeto”.
− Com o módulo principal selecionado no painel *Hierarchy* expanda a opção *User Constraints* do painel *Processes* clicando no símbolo + ao lado dela.
− Clique 2 vezes sobre a opção I/O Pin Planning (PlanAhead) - Pre-Synthesis
− Aguarde a inicialização do programa PlanAhead e feche a janela *Welcome* (se ela for apresentada).
− No painel *I/O Ports*, clique 2 vezes sobre a linha *Scalar ports* (destacada na parte A da figura a seguir) para que sejam exibidas as entradas e saídas como na figura.
− No mesmo painel *I/O Ports*, preencha a coluna *Site* (destacada como B na figura a seguir) digitando o pino associado a cada entrada e saída do circuito (os pinos são apresentados em outra figura). Sempre pressione a tecla *enter* após a inserção. Caso o sinal seja um vetor, pode ser necessário expandi-lo clicando 2 vezes sobre o nome do sinal.



A figura a seguir apresenta os principais dispositivos de entrada e saída da placa utilizada no laboratório. No retângulo colorido são apresentados as identificações dos pinos do FPGA conectados a esses dispositivos. Observe que os LEDs LD4 a LD7 apresentam 2 pinos e, no nosso caso, devemos usar o pino destacado com asterisco.



− Salve as modificações e feche o *PlanAhead*.

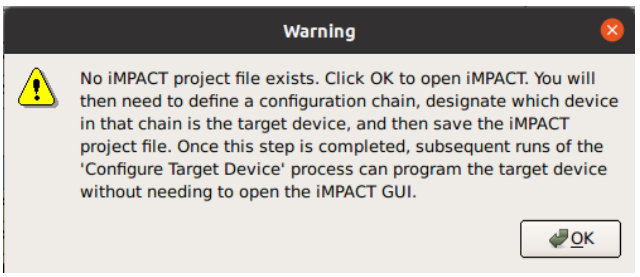
Gerar *bitstream*

− Verifique se o módulo principal do projeto está correto (no painel *Hierarchy* o símbolo fica a frente deste módulo). Caso esteja incorreto, faça o procedimento da seção “Trocar o módulo principal do projeto”.
− Se você ainda não associou os pinos às entradas e saídas do circuito, faça o procedimento da seção “Associar os pinos do FPGA às entradas e saídas”.
− No painel *Processes*, clique 2 vezes sobre a opção *Generate Programming File*.
− Caso o arquivo seja gerado, aparecerá um ícone verde ao lado da opção. Caso contrário, aparecerá um ícone vermelho. Verifique a aba console para verificar o erro. Corrija-o e repita a operação.

Configurar FPGA

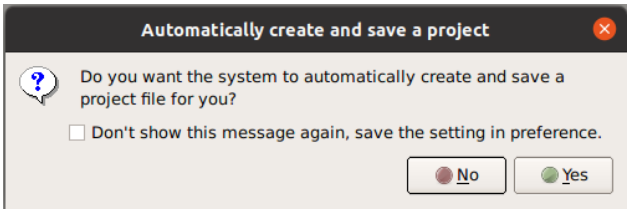
1. Se você ainda não tenha gerado o *bitstream*, faça o procedimento da seção “Gerar bitstream”.
2. Conecte o cabo USB à placa e ao computador.

3. Ligue a placa utilizando a chave “POWER”.
4. Clique duas vezes sobre a opção *Configure target device* no painel *Processes*. Caso seja gerada uma janela de notificação como da imagem a seguir, clique no botão OK.

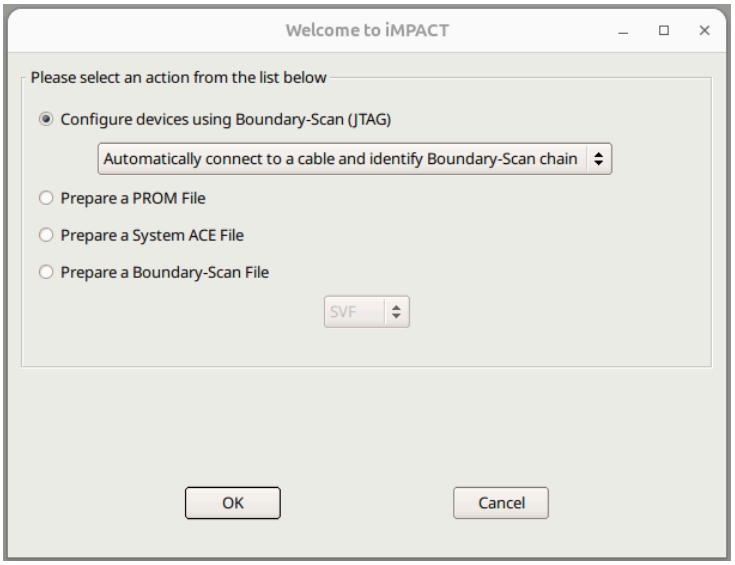


A operação anterior abrirá o software ISE IMPACT.

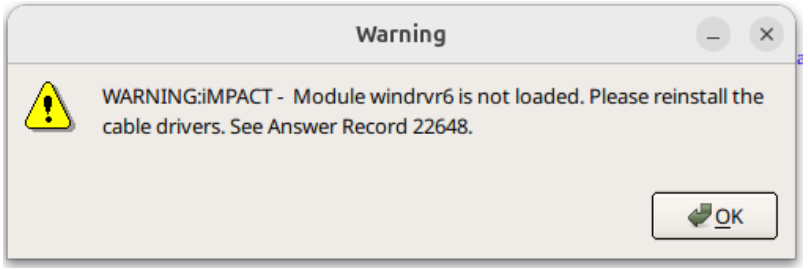
5. No menu, clique em File → New Project. Caso apareça a janela apresentada na figura a seguir, clique no botão Yes.

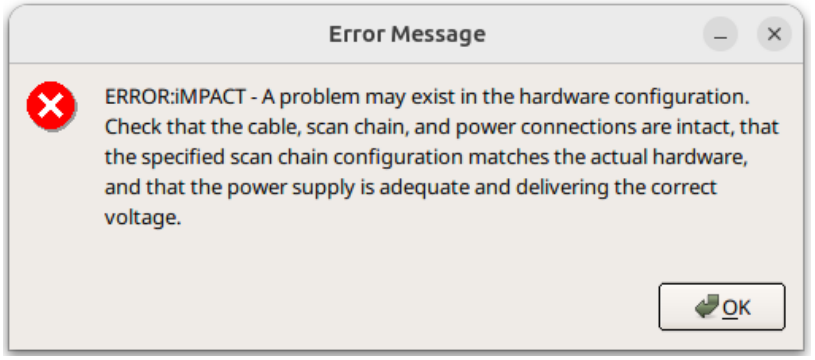


6. Na janela Welcome to iMPACT, como apresentado na figura a seguir, com as opções “Configure devices using Boundary-Scan (JTAG)” e “Automatically connect to a cable and identify Boundary-Scan chain” selecionadas, clique no botão OK.

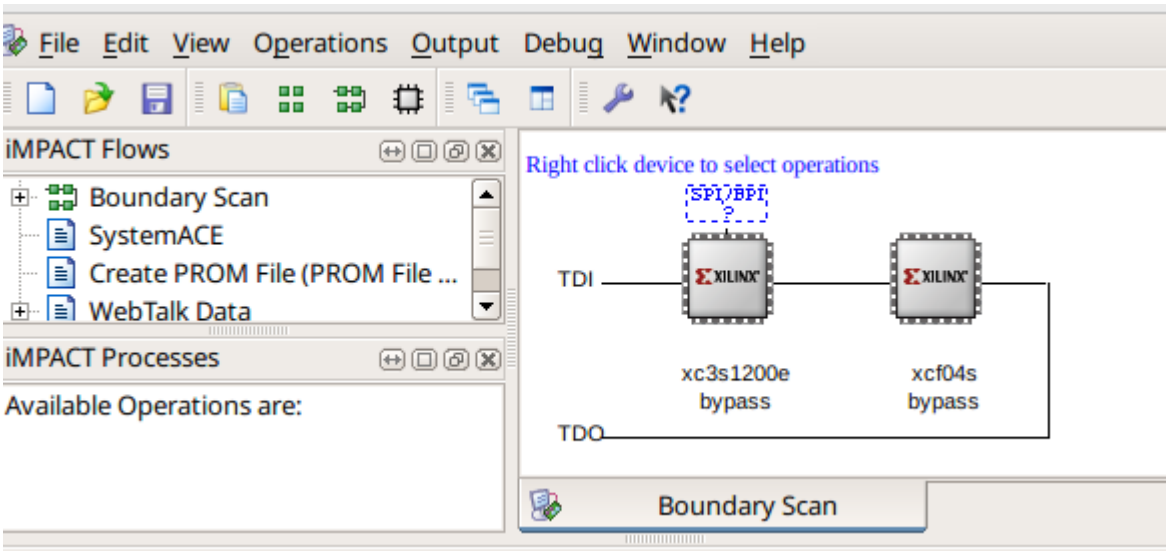


Nesse momento, o software tentará reconhecer a placa e o FPGA. Caso seja exibida alguma das janelas a seguir, você teve um problema. A primeira janela é exibida quando o cabo USB não está conectado, ou está em uma porta USB com mau funcionamento ou o *driver* da placa não foi configurado (verifique a seção *Configurar driver da placa*). Caso seja exibida a segunda janela, a placa está conectada, mas não está com a chave POWER ligada. Corrija o erro e faça o passo 5 novamente.

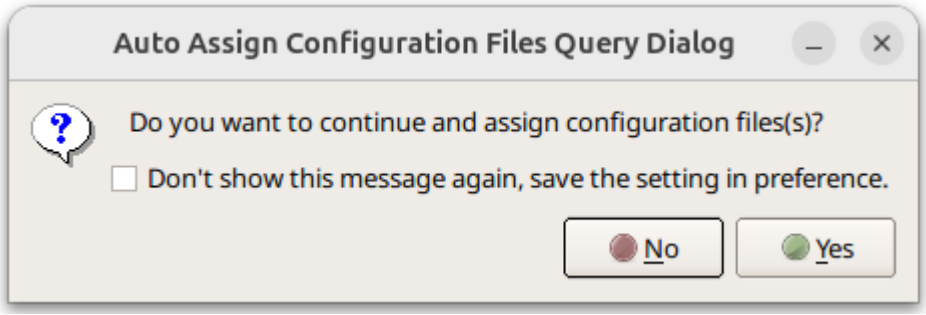




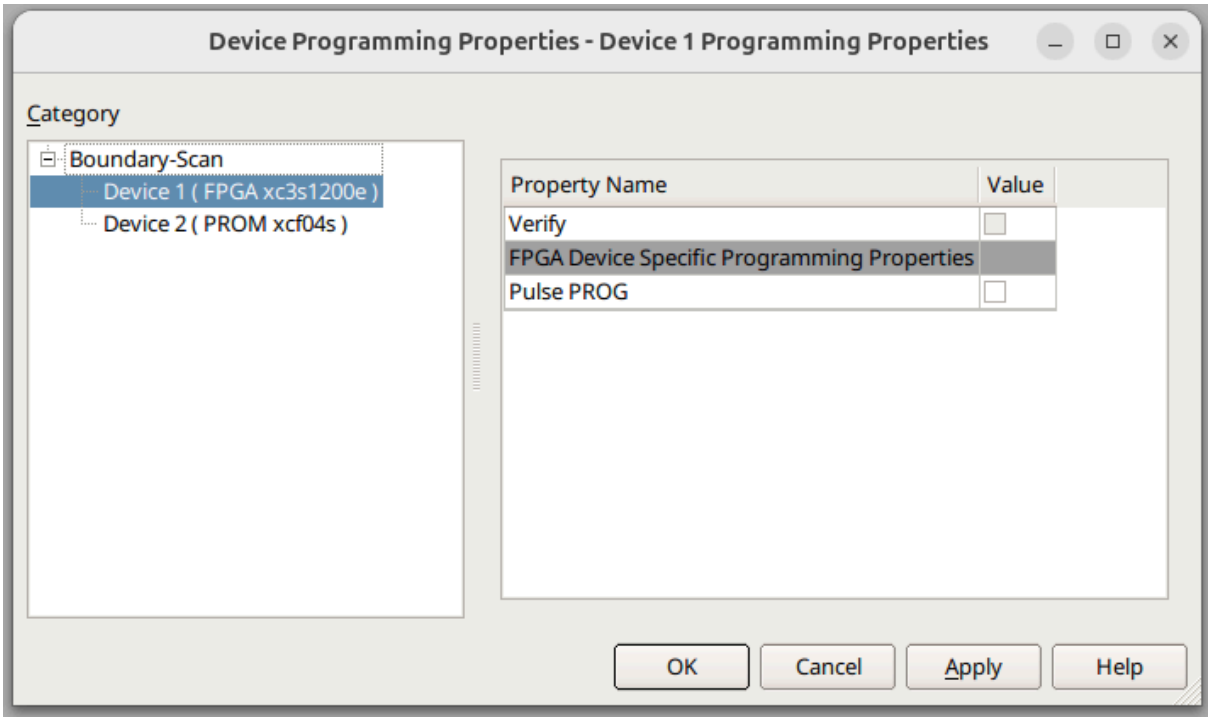
Caso o passo anterior tenha reconhecido o FPGA com sucesso, aparecerá na tela uma representação dos 2 dispositivos que permitem configuração na placa, como mostrado na figura a seguir.



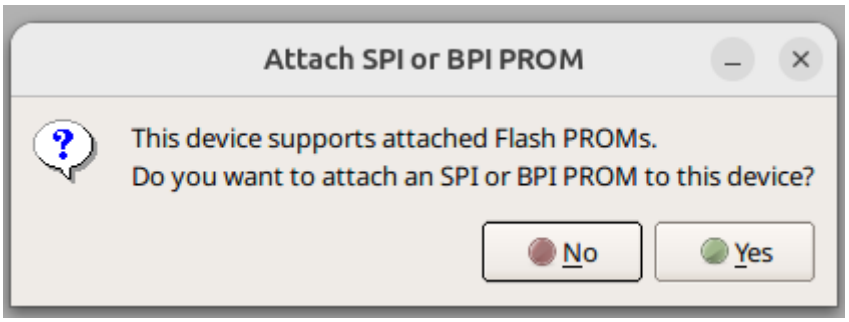
Caso a janela "*Auto Assign New Configuration File*", mostrada a seguir, seja exibida, clique em "No".



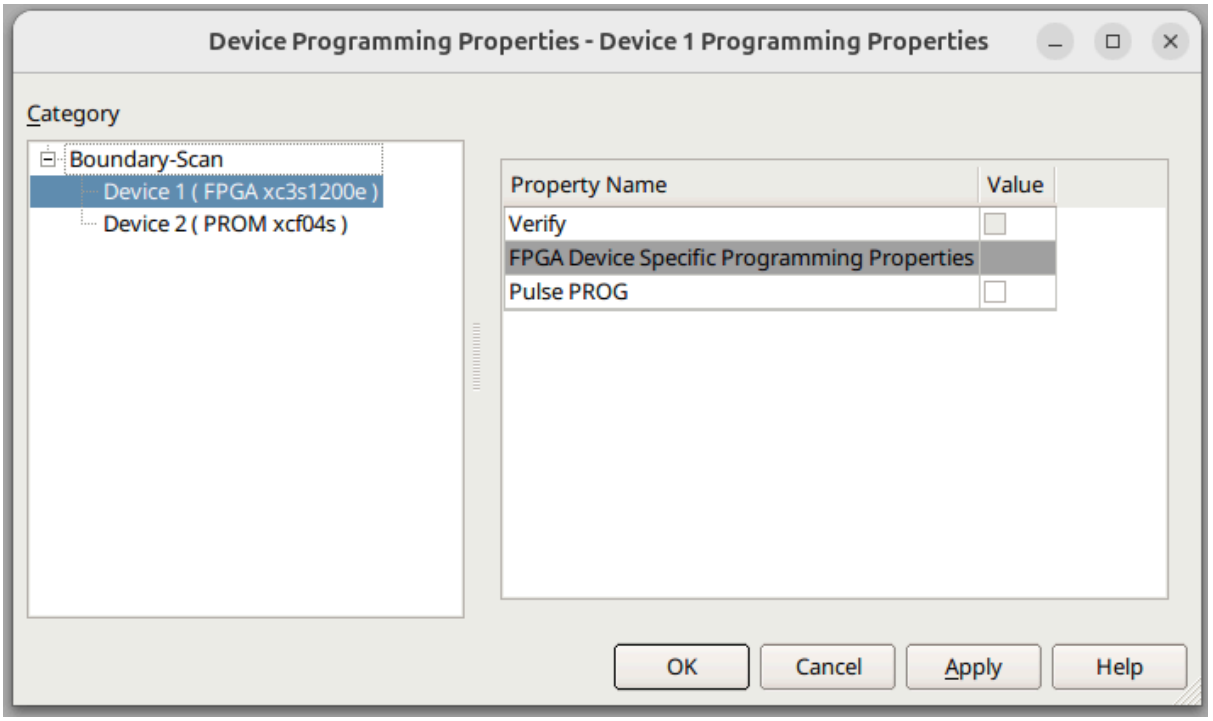
Caso a janela "*Device Programming Properties*", mostrada a seguir, seja exibida, clique em "Ok".



7. Clique com o botão direito do *mouse* sobre o dispositivo identificado como xc3s1200 e selecione a opção "*Assign New Configuration File*". Verifique a pasta do projeto, selecione o arquivo que contenha o *bitstream* (nome_do_circuito.bit) e clique no botão "*Open*".
   Caso a janela a seguir seja apresentada, perguntando se você quer configurar o outro dispositivo, a memória Flash, clique no botão "*No*".



8. Clique com o botão direito do *mouse* sobre o dispositivo identificado como xc3s1200 novamente e selecione a opção "*Program*".
   Caso a janela "*Device Programming Properties*", mostrada a seguir, seja exibida, clique em "Ok".



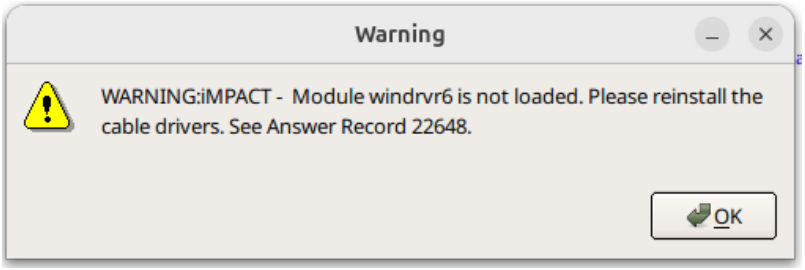
Quando a mensagem “*Program succeeded*” for apresentada, o LED Done acenderá na plana e você já poderá testar seu circuito.

Visualizar diagrama esquemático correspondente à descrição em VHDL

− Verifique se o módulo que você quer visualizar está marcado como o principal do projeto (no painel *Hierarchy* o símbolo fica a frente deste módulo). Caso esteja incorreto, faça o procedimento da seção “Trocar o módulo principal do projeto”.
− Clique no sinal de + ao lado de Synthesize - XST.
− Clique 2 vezes em "View RTL Schematic".
− Caso seja exibida uma janela com opções, deixe marcada a opção “Start with a schematic of the top-level block” e clique OK.
− Clique duas vezes sobre a caixa do circuito exibido ou com o botão direito sobre a caixa e clique em "Show Block Contents" para visualizar seu interior.

Configurar *driver* da placa

Se você está recebendo o erro da figura a seguir quando utiliza o Impact para configurar o FPGA, pode ser que o *driver* da placa não esteja instalado no usuário que você está logado. Outras causas são: o cabo USB não está conectado ou está em uma porta USB com mau funcionamento.



Para configurar o driver da placa realize o procedimento a seguir:

1. Baixe o arquivo ponto-cse.tar.gz clicando no link e depois no ícone  .
2. Na pasta Downloads, descompacte o arquivo clicando sobre ele 2 vezes. Isso fará com que uma pasta chamada ponto-cse seja criada.
3. Dentro da pasta ponto-cse, pressione as teclas Ctrl+h para que os arquivos ocultos sejam exibidos. Copie o conteúdo da pasta ponto-cse para a raiz da sua home.
4. Se estiver com o Impact aberto, fecho-o e abra novamente.

Restaurar *layout* do ISE

Caso algum painel ou aba “desapareça”, você pode voltar à formatação original de abas e painéis clicando no menu *Layout →Load Default Layout*.

VHDL

Estrutura básica de um código em VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity nome_circuito is
     --definição da entidade
end nome_circuito;

architecture nome_arquitetura of nome_circuito is
begin
     --definição da arquitetura
end nome_arquitetura;
```

Implementação de uma entidade

Definição genérica:

```
entity nome_da_entidade is
     port ($lista_de_sinais);
end nome_da_entidade;
```

Especificidades:

`nome_da_entidade` é o nome escolhido pelo desenvolvedor para identificar o circuito, formado por caracteres alfanuméricos e *underline*, sendo que o primeiro caractere deve ser uma letra. Este deve ser o mesmo nome do arquivo com extensão .vhd.

`$lista_de_sinais` é a lista de entradas e saídas da entidade definida como:

```
nome_do_sinal1 : $modo $tipo_do_sinal; ...
nome_do_sinalN : $modo $tipo_do_sinal
```

`nome_do_sinal` é o nome escolhido pelo desenvolvedor para identificar a porta, formado por caracteres alfanuméricos e *underline*, sendo que o primeiro caractere deve ser uma letra.

`$modo` especifica a "direção" da porta:

- `in`: entrada da entidade
- `out`: saída da entidade
- `inout`: entrada e saída da entidade (bidirecional)
- `buffer`: saída da entidade com registro

$tipo_do_sinal é um tipo de sinal padrão ou definido pelo usuário (usaremos o tipo std_logic e seus derivados)

Cada porta de entrada e saída é vista como um sinal dentro da arquitetura do módulo.

Implementação da arquitetura de uma entidade

```
architecture nome_da_arquitetura of  $nome_da_entidade is
-- Parte declarativa da arquitetura
     $declaração_de_sinal
begin
-- Parte de descrição do módulo
     $descrição_do_módulo
end nome_da_arquitetura;
```

Declaração de sinais

Deve ser feita na parte declarativa da arquitetura.

```
signal nome_do_sinal : tipo_do_sinal;
```

Atribuição de valores a saídas e sinais

Deve ser feita na parte de descrição do módulo na arquitetura.

```
nome_do_sinal <= valor;
```

Valor pode ser uma constante, um sinal, uma entrada ou uma expressão lógica. Exemplos:

```
A <= '1';  -- constante
A <= B;    -- sinal
A <= (NOT B AND C) OR D;   -- expressão lógica
```

Tipo std_logic

Valores:

- 'U': sem inicialização
- 'X': valor desconhecido
- '0': nível lógico baixo
- '1': nível lógico alto
- 'Z': alta impedância
- '-': *don't care*
- 'W': sinal fraco, não é possível afirmar que seja 0 ou 1
- 'L': sinal fraco, provavelmente é 0
- 'H': sinal fraco, provavelmente é 1

Alguns desses valores não podem ser utilizados para síntese, somente para simulações.

Expressões lógicas

Expressões formadas pelos operadores `and, or, nand, nor, xor, xnor` e `not`.
Parênteses devem ser utilizados para garantir a prioridade dos operadores.

Declaração de componentes

Deve ser feita na parte declarativa da arquitetura. É muito parecida com a declaração da entidade do circuito utilizado. Deve ser feita uma única vez em cada arquitetura independentemente do número de instâncias utilizadas.

```
component nome_do_componente is
     port($lista_de_sinais);
end component;
```

$lista_de_sinais é igual à lista de sinais feita na declaração da entidade (port)

Instanciação de componente

Realizada no corpo da arquitetura.
Para usar um componente, a arquitetura deve declarar o mesmo para depois instanciá-lo.
Podem existir inúmeras instâncias de um componente em uma arquitetura.

```
etiqueta_de_instanciação : nome_do_componente
      port map($lista_de_mapeamento_de_portas);
```

$lista_de_mapeamento_de_portas é um conjunto de atribuições de portas do componente para sinais da arquitetura.

porta1_componente => sinal_interno_à_arquitetura,
...,
portaN_componente => sinal_interno_à_arquitetura

A direção da “seta” é a mesma independente da porta ser de entrada ou saída

Tipo std_logic_vector

Pertencente ao pacote IEEE 1164.
Arranjo unidimensional (vetor) de elementos do tipo std_logic.
Na declaração do sinal indica-se o intervalo de índices e se a indexação é crescente ou decrescente.

Exemplo considerando que M > N:

```
signal s0 : std_logic_vector(N to M);
signal s1 : std_logic_vector(M downto N);
```

Constantes do tipo std_logic_vector são escritas entre aspas duplas, em binário, octal ou hexadecimal.

- Cada dígito octal corresponde a 3 elementos do vetor
- Cada dígito hexadecimal corresponde a 4 elementos vetor

Devem conter o número exato de elementos do vetor. Exemplos:

```
signal s0 : std_logic_vector(5 downto 0);
signal s1 : std_logic_vector(7 downto 0);
s0 <= “001100”;            -- binário
s1 <= B“00001100”;    -- binário
s0 <= O“14”;          -- octal
s1 <= X“0C";          -- hexadecimal
```

Acesso aos elementos do vetor

- Acesso a um elemento: utilização do nome do vetor e do índice entre parênteses
- Acesso a todo o vetor: utilização do nome do vetor
- Acesso a parte do vetor: utilização do nome do vetor e do intervalo entre parênteses

Exemplos:

```
signal s0, s1 : std_logic_vector(3 downto 0);
s0(3) <= ‘1’;              -- acesso a um elemento
s1 <= s0;                  -- acesso a todo o vetor
```

```
s0(2 downto 0) <= "100";   -- acesso a parte do vetor
```

Processo

Bloco em que é possível descrever um módulo (ou parte dele) de maneira comportamental, semelhante à forma que utilizamos quando desenvolvemos um programa em linguagem de alto nível.
Um processo deve ser descrito na parte de descrição da arquitetura e é executado, normalmente, em resposta à mudança de valores de sinais especificados em sua definição, na lista sensível.

```
id_do_processo : process($lista_sensível)
     -- parte declarativa do processo
begin
     -- corpo do processo
end process id_do_processo;
```

O processo é executado toda vez que algum sinal da lista sensível tem seu valor alterado. Nessa lista podem ser colocados os sinais declarados na arquitetura ou as entradas da entidade. Em um *testbench* pode ser utilizado um processo sem lista sensível, indicando que este processo será executado para sempre, a partir do início da simulação, a não ser que um comando `wait` seja executado.
É possível declarar variáveis em um processo e elas serão visíveis somente em seu interior. As variáveis devem ser declaradas na parte declarativa do processo.
No corpo do processo, devem ser colocadas instruções que serão executadas sequencialmente. Existem diversas classes de instruções, como instruções de repetição, condicionais, de atribuição, etc. No entanto, devemos sempre lembrar que o comportamento descrito será sintetizado em hardware e, portanto, não se comporta como em um programa descrito em uma linguagem de programação de alto nível.

Variáveis

Somente podem ser utilizadas em processos.
Declaração (feita na parte declarativa do processo):

```
variable nome_da_variavel : tipo_da_variavel;
```

Atribuição (feita no corpo do processo):

```
nome_da_variavel := valor;
```

Valor pode ser uma constante, uma variável, um sinal, uma entrada ou uma expressão lógica.

Instruções sequenciais condicionais

```
if condição then
```

```
     -- comandos
end if;

if condição then
     -- comandos
else
     -- comandos
end if;

if condição then
     -- comandos
elsif condição then
     -- comandos
end if;
```

As condições são definidas com comparadores lógicos.

Comparadores lógicos de igualdade/desigualdade

Para comparar se 2 valores std_logic são iguais ou não, você deve utilizar o comparador = ou /=.

Laços de repetição

```
for contagem loop
     comandos que serão repetidos
end loop;
```

A contagem pode ser realizada de duas maneiras: crescente ou decrescente. Ela sempre deve indicar um contador e o contador pode ser utilizado como uma variável inteira no corpo do for.
Exemplo com contagem crescente:

```
     for i in 0 to 5 loop
```

Exemplo com contagem decrescente:

```
     for i in 5 downto 0 loop
```

Generate

Exemplo de matriz de registradores de 4 bits onde cada registrador é uma instância do mesmo componente:

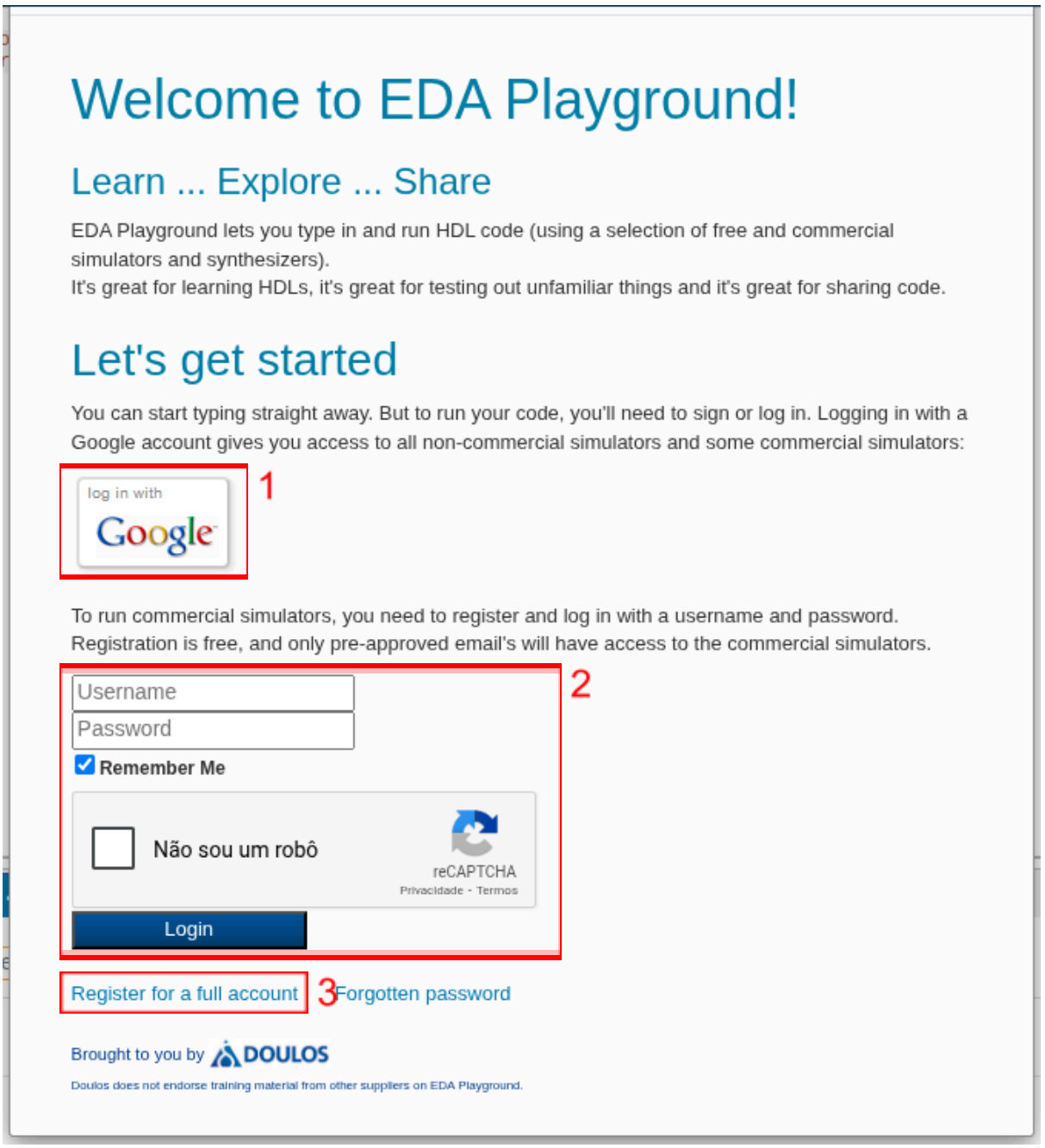
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RegisterArray is
  port (
    Clock: in STD_LOGIC;
    DataIn: in STD_LOGIC_VECTOR(3 downto 0);
```

```
    Enable: in STD_LOGIC;
    DataOut: out STD_LOGIC_VECTOR(3 downto 0)
  );
end RegisterArray;

architecture Behavioral of RegisterArray is
  component Register is
    port (
      Clock: in STD_LOGIC;
      DataIn: in STD_LOGIC_VECTOR(3 downto 0);
      Enable: in STD_LOGIC;
      DataOut: out STD_LOGIC_VECTOR(3 downto 0)
    );
  end component;

begin
  -- Usando 'for generate' para criar uma matriz de registradores
  gen_registers: for i in 0 to 3 generate
    reg_inst: Register
      port map (
        Clock => Clock,
        DataIn => DataIn,
        Enable => Enable,
        DataOut => DataOut(i * 4 + 3 downto i * 4)
      );
  end generate gen_registers;
end Behavioral;
```

EDA Playground

Acesse o EDA Playground pelo link https://www.edaplayground.com/

Antes de usar o EDA Playground, é indicado que você crie uma conta com seu e-mail da UFSJ para ter acesso a todas as ferramentas disponíveis de maneira gratuita. Caso você opte por acessar com uma conta Google, você terá acesso limitado a algumas ferramentas e não conseguirá simular circuitos sequenciais.
Na figura a seguir é apresentada a tela inicial da ferramenta. Clique no botão destacado como 1 se quiser fazer login com a conta Google, ou preencha seus dados na região destacada como 2, ou clique no link destacado como 3 para criar uma conta.



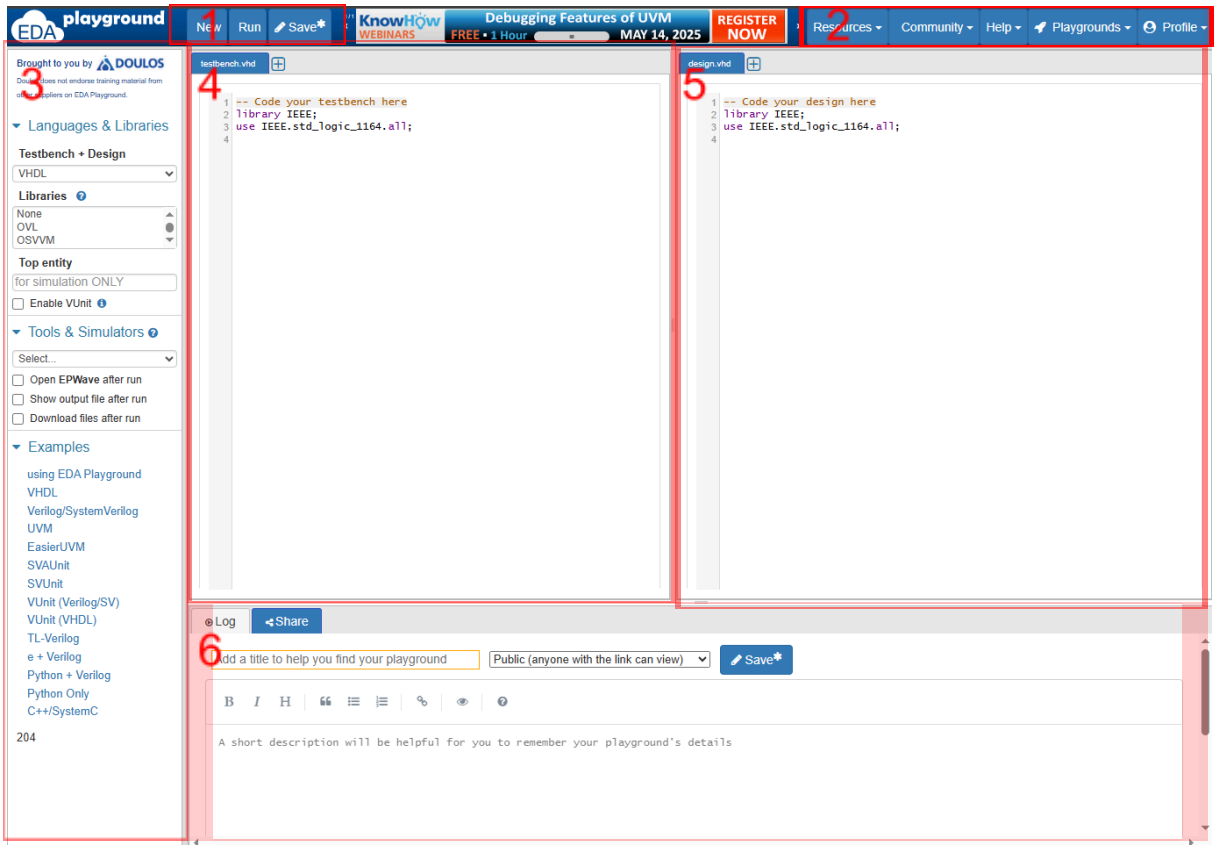
Tela de projeto no EDA Playground

A tela de projeto do EDA Playground é organizada em painéis, descritos a seguir. A figura a seguir destaca cada um deles.

Painel 1: Contém os botões para criar um novo projeto (New), executar o projeto corrente (Run) e salvar as modificações realizadas no projeto (Save). Lembre-se de sempre clicar sobre o botão Save, pois as alterações **não** são salvas automaticamente.

Painel 2: Contém botões que dão acesso aos recursos disponibilizados pela empresa Doulos (*Resources*, *Community* e *Help*), aos seus projetos e de outras pessoas que publicaram projetos (*Playgrounds*) e às suas configurações pessoais (*Profile*).

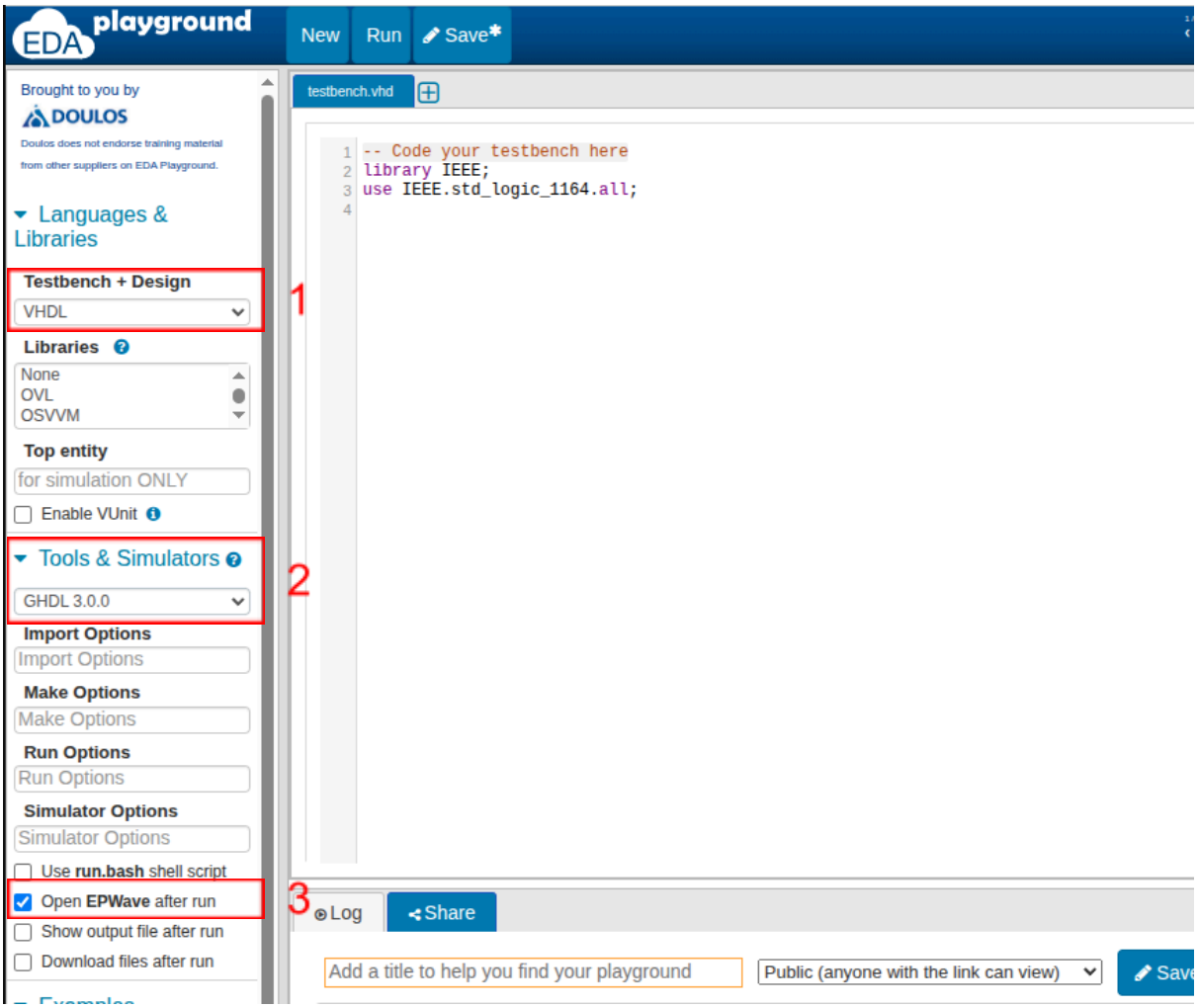
Painel 3: Esse é o painel de configurações do projeto. Você pode ver mais detalhes dele na seção “Configuração do projeto no EDA Playground”.
Painel 4: Nesse painel devem ser colocados todos os arquivos de *testbench* do projeto. A entidade que será simulada deve ser identificada (sem .vhd) no campo “*Top entity*” do painel de configurações do projeto.
Painel 5: Nesse painel devem ser colocados todos os arquivos de circuitos e componentes do projeto. Ao criar um novo arquivo, lembre-se de colocar a extensão .vhd.
Painel 6: Esse painel contém 2 abas, *Log* e *Share*, sendo visível a aba que está em azul. A aba *Log* apresenta o log da execução da simulação e os possíveis erros encontrados nas verificações dos circuitos, componentes e *testbenchs*. A aba *Share*, permite que você dê um nome/título e uma descrição ao projeto, além de configurar as permissões de acesso ao seu projeto: *Public* (qualquer pessoa com o *link* do projeto pode visualizá-lo), *Published* (o projeto é compartilhado na seção *Published playgrounds* pela comunidade) e *Private* (o projeto só pode ser acessado por você). Depois de salvar o projeto pela primeira vez, também são apresentados os botões para compartilhamento do projeto. O botão com uma corrente permite que você copie o *link* do projeto.



Configuração do projeto no EDA Playground

Os passos a seguir devem ser feitos nessa sequência. Na barra lateral, na opção “*Testbench + Design*”, destacada na figura a seguir como 1, selecione VHDL. Clique em “*Tools & Simulators*” para expandir essa seção e selecione GHDL 3.0.0 na caixa destacada na figura como 2, no entanto, eu não consegui simular circuitos sequenciais com ele. Caso você esteja utilizando uma conta da UFSJ, você poderá utilizar a opção Aldec Riviera Pro

2023.04. Deixe marcada a opção “*Open EPWave after run*”, destacada com o número 3, na figura. Coloque seu módulo principal na aba design.vhd e seu testbench na aba testbench.vhd. Caso você tenha subcircuitos, você deverá clicar no símbolo + ao lado da aba design.vhd e clique no botão “Upload files…” caso você já tenha o arquivo pronto ou insira um nome na caixa “Filename”, com a extensão .vhd, e clique no botão “Create file”. Coloque na caixa “Top entity” o nome da entidade do seu testbench.



Simulação do projeto no EDA Playground

Caso os sinais não apareçam na simulação, clicar no botão “Get Signals” e selecionar os sinais que devem ser observados.