



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Lab 5

Criação de um Projeto no Vivado

1. INTRODUÇÃO

Para trabalhar e modificar o Sistema RVfpga, terá de criar um projeto que inclua todo o Verilog, SystemVerilog, configuração, e ficheiros que definem o sistema. Neste Lab, mostramos como criar um projeto Vivado que visa o SoC utilizado neste curso para a placa Nexys A7 FPGA da Digilent, -100T versão (ou seja, RVfpgaNexys). (Lembre-se que se já tiver uma placa Nexys4 DDR, também pode usá-la). Ao seguir estes mesmos passos, poderá modificar RVfpgaNexys e resintetizá-lo.

IMPORTANTE: Se ainda não tiver feito isso, instale o Vivado 2019.2 WebPACK da Xilinx, conforme explicado no Guia de Introdução.

Utilizará o Vivado Design Suite¹ da Xilinx para construir o sistema RVfpgaNexys utilizando o RTL, os ficheiros Verilog que definem o sistema. Siga estes passos, detalhados abaixo, para construir o sistema RVfpgaNexys para a placa Nexys A7 FPGA.

Passo 1. Abrir o Vivado

Passo 2. Criar um novo projeto RTL

Passo 3. Adicionar os ficheiros fonte RTL e os ficheiros de condicionantes

Passo 4. Selecionar Nexys A7 como placa alvo

Passo 5. Definir rvfpganexys como "Top Module" e common_defines.vh como global

Passo 6. Gerar o ficheiro de configuração (Bitstream)

Passo 1. Abrir o Vivado

Se não instalou o Vivado na sua máquina como descrito no Guia de Introdução ao RVfpga, faça-o agora. Certifique-se de que instala também os ficheiros da placa FPGA.

Agora, corra o Vivado (em **Linux**, abra um terminal e escreva: vivado; em **Windows**, abra o Vivado a partir do menu Iniciar). O ecrã de boas-vindas do Vivado irá abrir. Clique em Criar Projeto (ver Figura 1).

¹ Nestes materiais utilizamos o Vivado 2019.2. Embora a maioria das coisas também deva funcionar em distribuições mais recentes do Vivado, recomendamos vivamente a utilização desta versão.

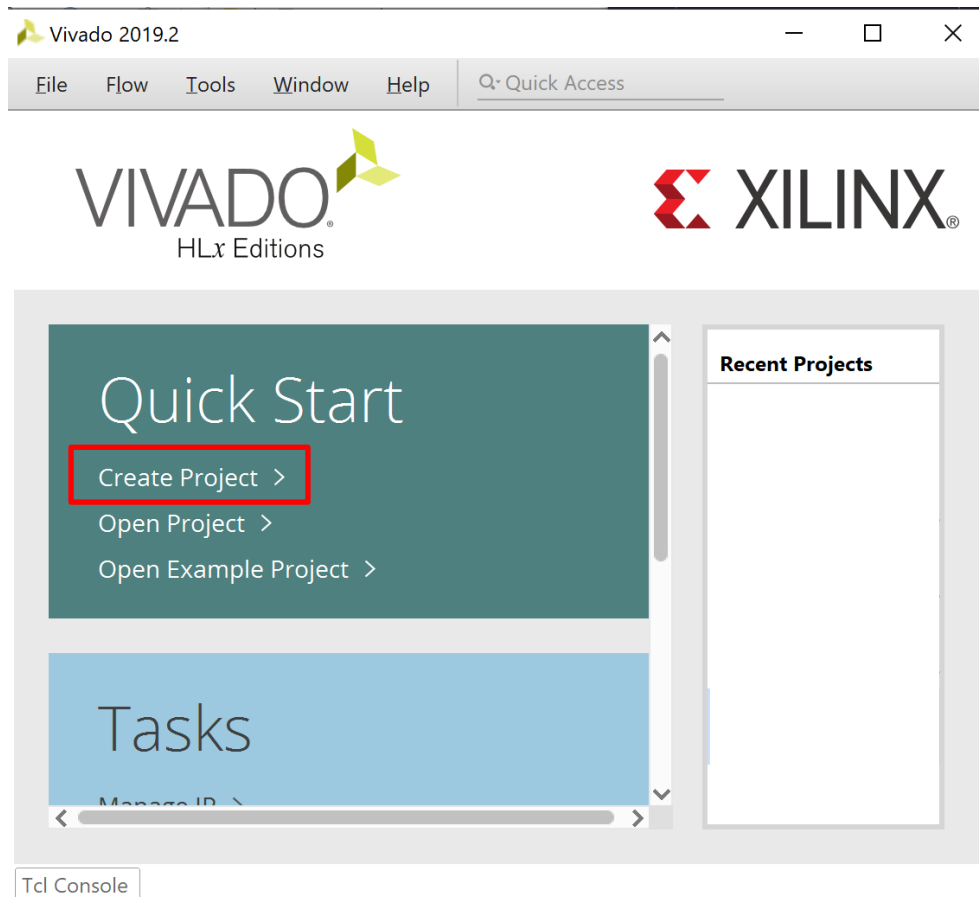


Figura 1. Ecrã de boas-vindas do Vivado: Criar Projeto

Passo 2. Criar um projeto RTL

O Assistente do *Create a New Vivado Project* irá agora abrir (ver Figura 2). Clique em *Next*.

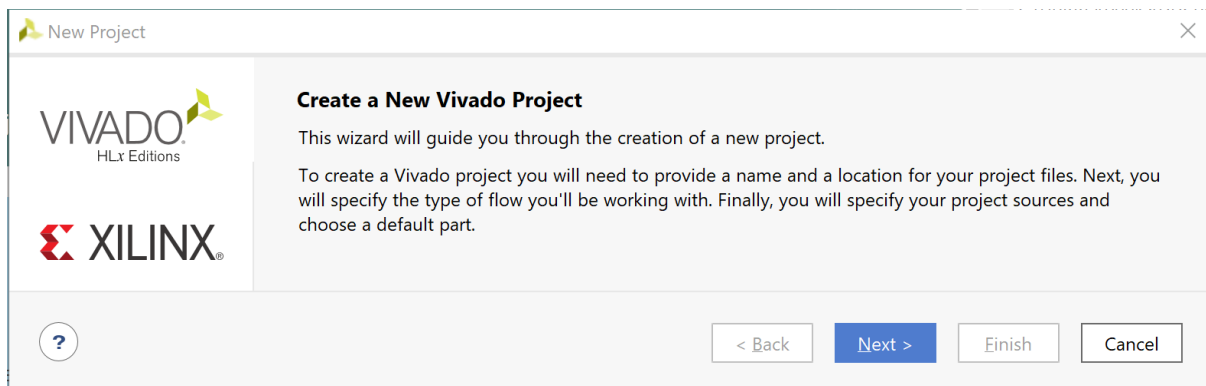


Figura 2. Assistente para criar um novo projeto Vivado (Project Wizard)

Dê ao projeto o nome “project_1” e coloque-o na pasta *[RVfpgaPath]/RVfpga/Labs/Lab1*. Selecione a opção *Create project subdirectory*. Em seguida clique em *Next* (ver Figura 3).

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

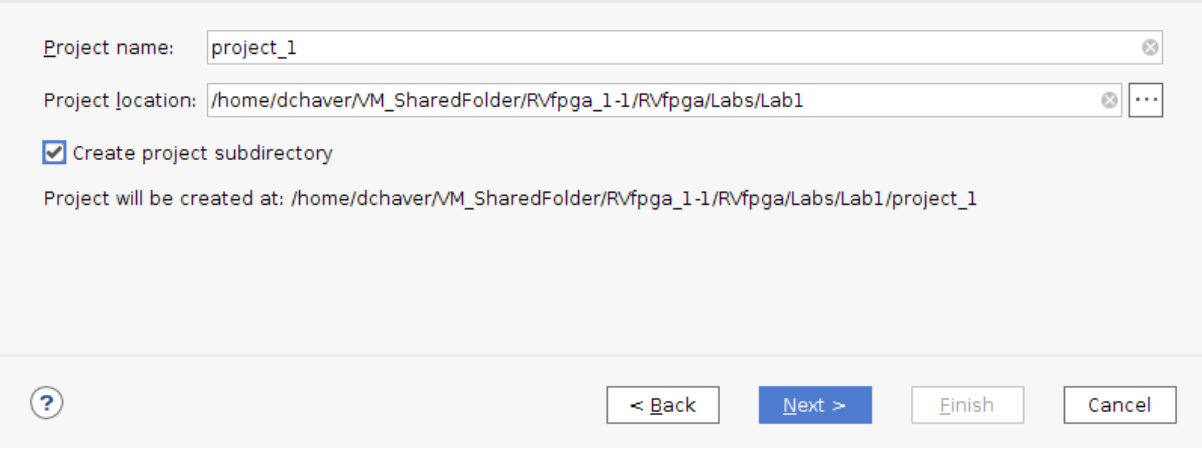



Figura 3. Nome do projeto

Selecione o tipo de projeto como Projeto RTL, e clique em *Next* (ver Figura 4).

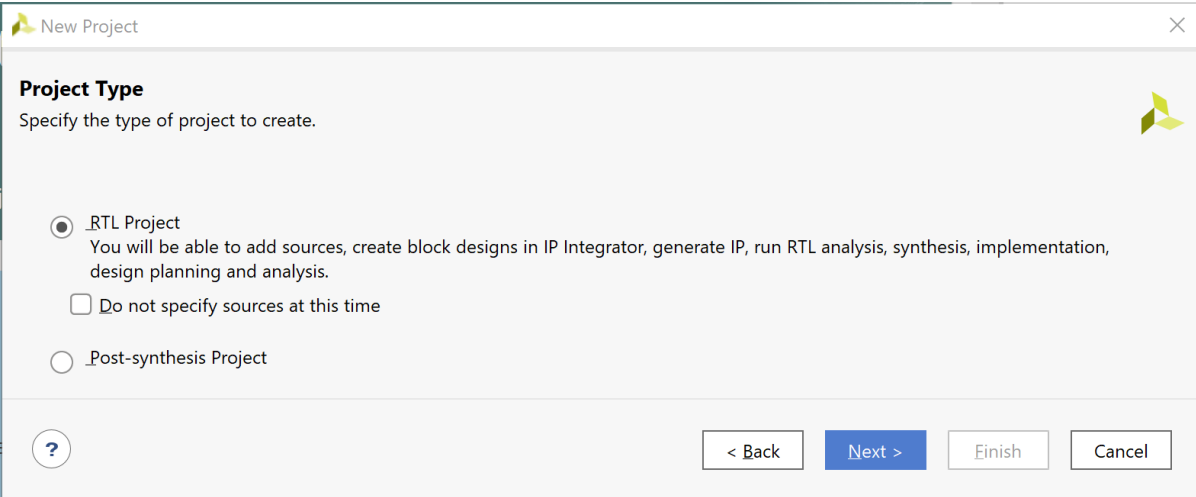


Figura 4. Projeto RTL

Passo 3. Adicionar os ficheiros fonte RTL e os ficheiros de condicionantes

Na janela *Add Sources*, clique em *Add Directories*, e selecione *[RVfpgaPath]/RVfpga/src* (ver Figura 5). Assegure-se de que ambas as opções seguintes são selecionadas (como mostrado na Figura 5):

- Procurar e adicionar ficheiros RTL ao projeto
- Adicionar ficheiros fonte de subdiretórios

Depois clique em *Next*.

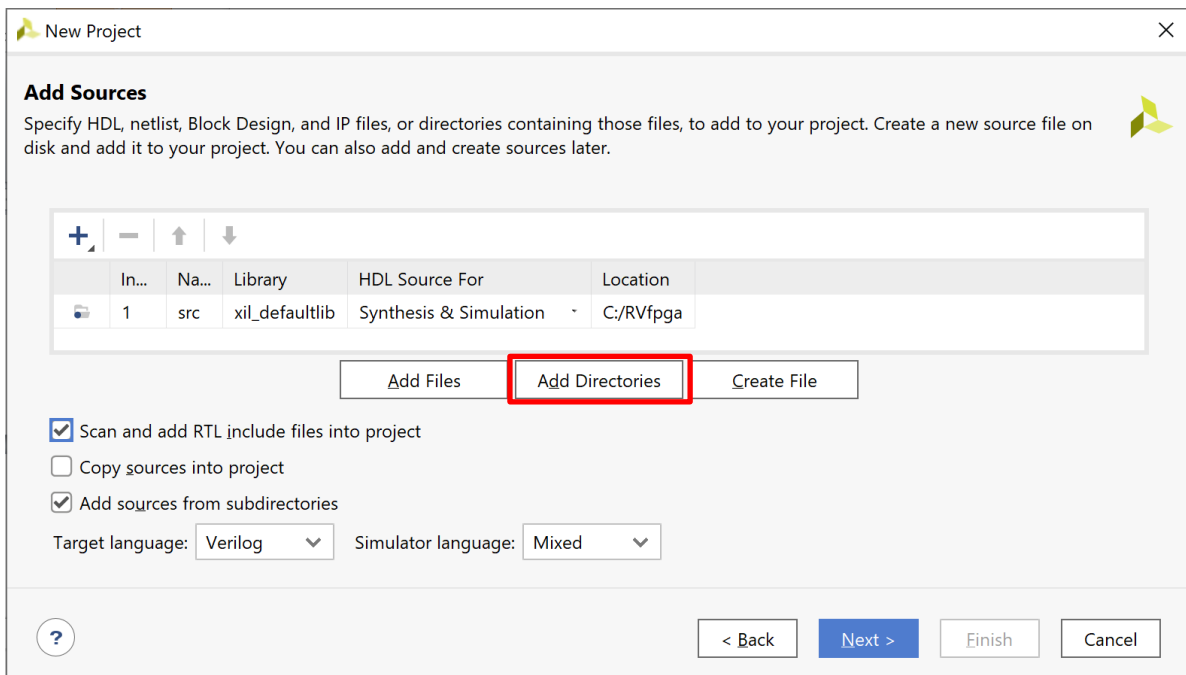


Figura 5. Adicionar Ficheiros Fonte

Agora irá adicionar as condicionantes ao sistema. Estes ficheiros mapeiam os nomes dos sinais para os pinos na placa. Por exemplo, os LEDs na placa Nexys A7 FPGA são ligados aos pinos da FPGA na placa através de pistas na PCB. O Vivado deve saber isto para que mapeie o nome correto do sinal no RTL para o pino FPGA correto. Por exemplo, a seguinte linha no ficheiro `[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc`, um ficheiro de *constraints* do projeto Xilinx, indica que o pino FPGA H17 mapeia para o bit menos significativo do LED (`o_led[0]`) e usa uma interface LVCMOS 3.3V:

```
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { o_led[0] }]
```

Note-se que o nome do sinal `o_led` é o nome usado no código Verilog para acionar os LEDs da placa Nexys A7.

Na janela *Add Constraints*, clique em *Add Files* e selecione os dois ficheiros seguintes (ver Figura 6):

```
[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc
[RVfpgaPath]/RVfpga/src/LiteDRAM/liteDRAM.xdc
```

Depois clique em *Next*.

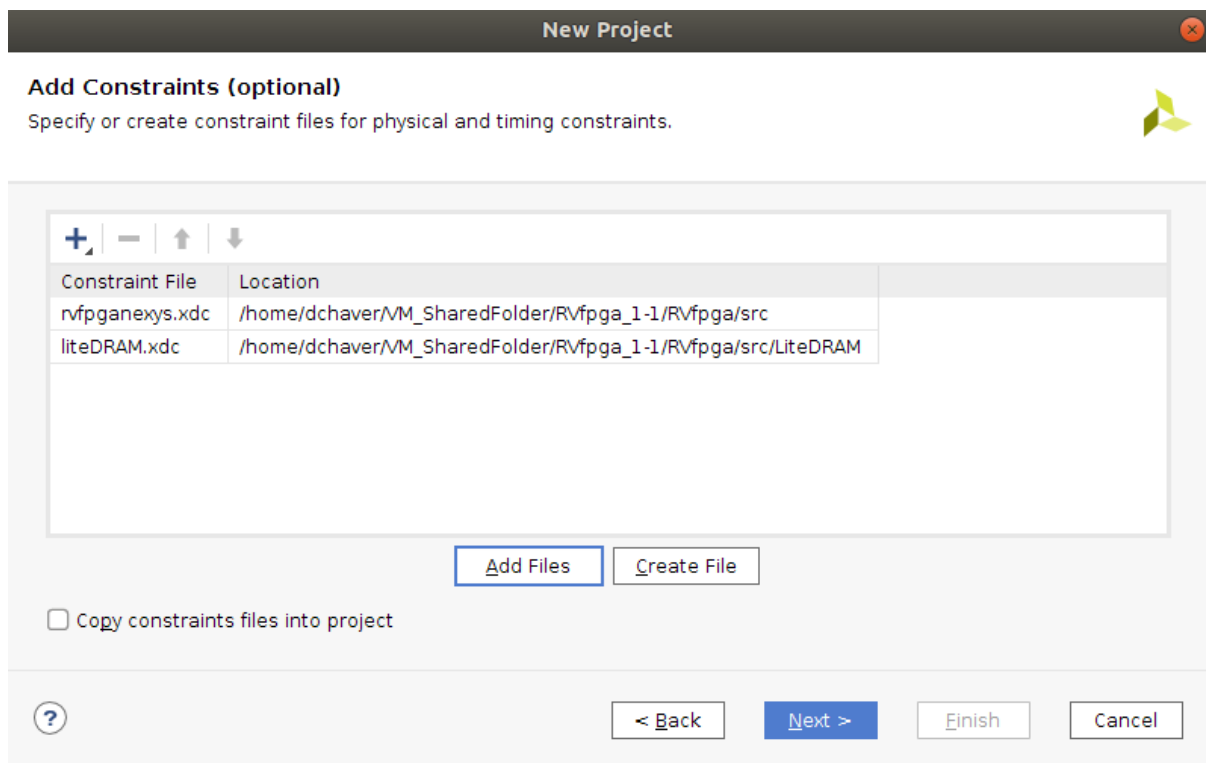


Figura 6. Adicionar Condicionantes

Passo 4. Selecionar Nexys A7 como placa alvo

Na janela *Default Part*, clique em *Boards* e depois selecione Nexys A7-100T (ver Figura 7). Pode utilizar a caixa de Pesquisa para restringir os resultados. Notará também que o nome do real da FPGA está listado na coluna *Part*: xc7a100tcsg324-1. Isto indica que é uma FPGA Xilinx Artix-7 com 100k portas equivalentes com um pacote CSG (*chip-scale grid*) e 324 pinos.

Clique em *Next*.

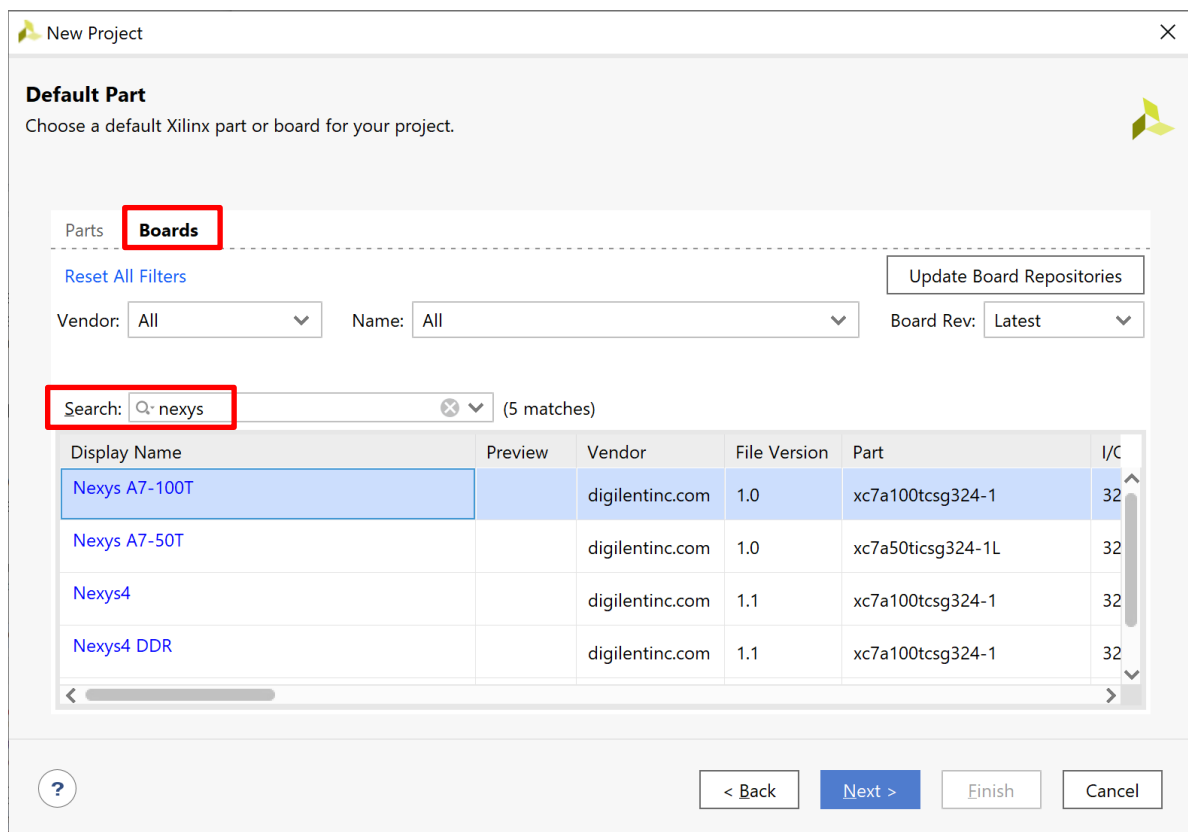


Figura 7. Selecionar a placa alvo: Nexys A7-100T

Na janela *New Project Summary*, clique em *Finish* (ver Figura 8).

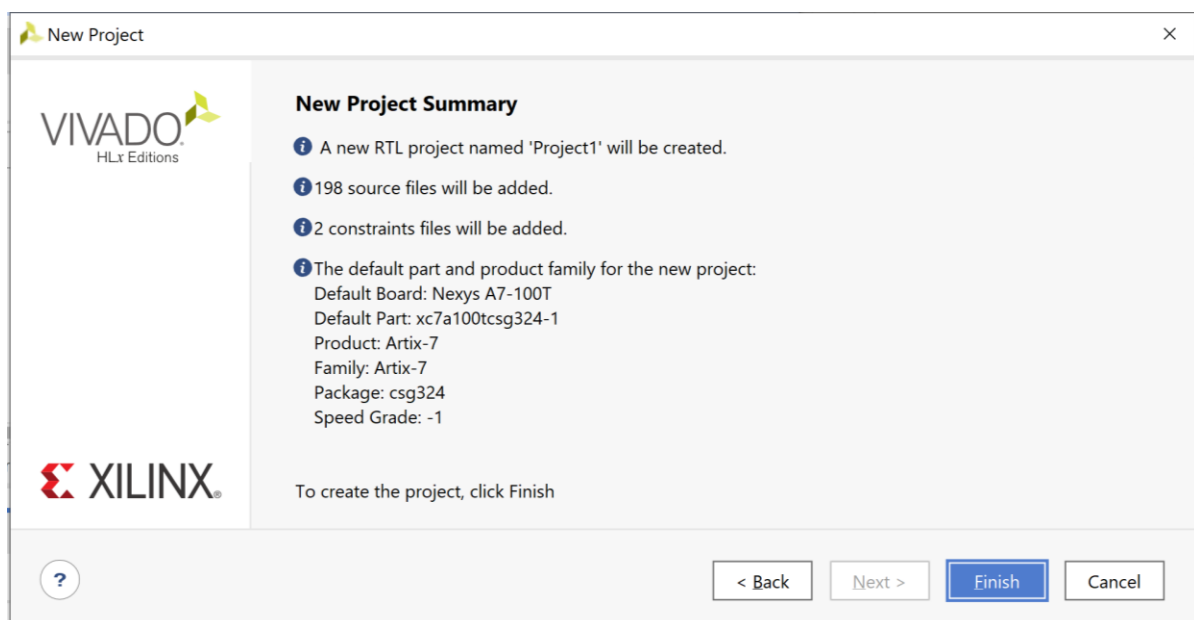


Figura 8. Janela “New Project Summary”

Note-se que, uma vez que o projeto esteja a ser configurado, indicará que existem ficheiros com erros de sintaxe - isto será corrigido na próxima etapa.

Passo 5. Configuração do projeto: Definir rvfpganexys como Módulo Principal, definir ficheiro common_defines.vh como global, adicionar boot_main.mem ao projeto e incluir pastas da Plataforma Pulp

Definir o rvfpganexys como Módulo de Topo: Agora você definirá o módulo *rvfpganexys* como o módulo principal. No painel *Sources*, desça até *Design Sources*, clique com o botão direito do mouse no módulo *rvfpganexys* e selecione *Set as Top* (ver Figura 9). Também é possível encontrar o módulo *rvfpganexys* escrevendo este nome na caixa de pesquisa, como mostrado. Isso define o *rvfpganexys* como o módulo de nível mais alto na hierarquia e o objeto a ser sintetizado e implementado no FPGA. Após definir *rvfpganexys* como o módulo superior, a hierarquia será atualizada.

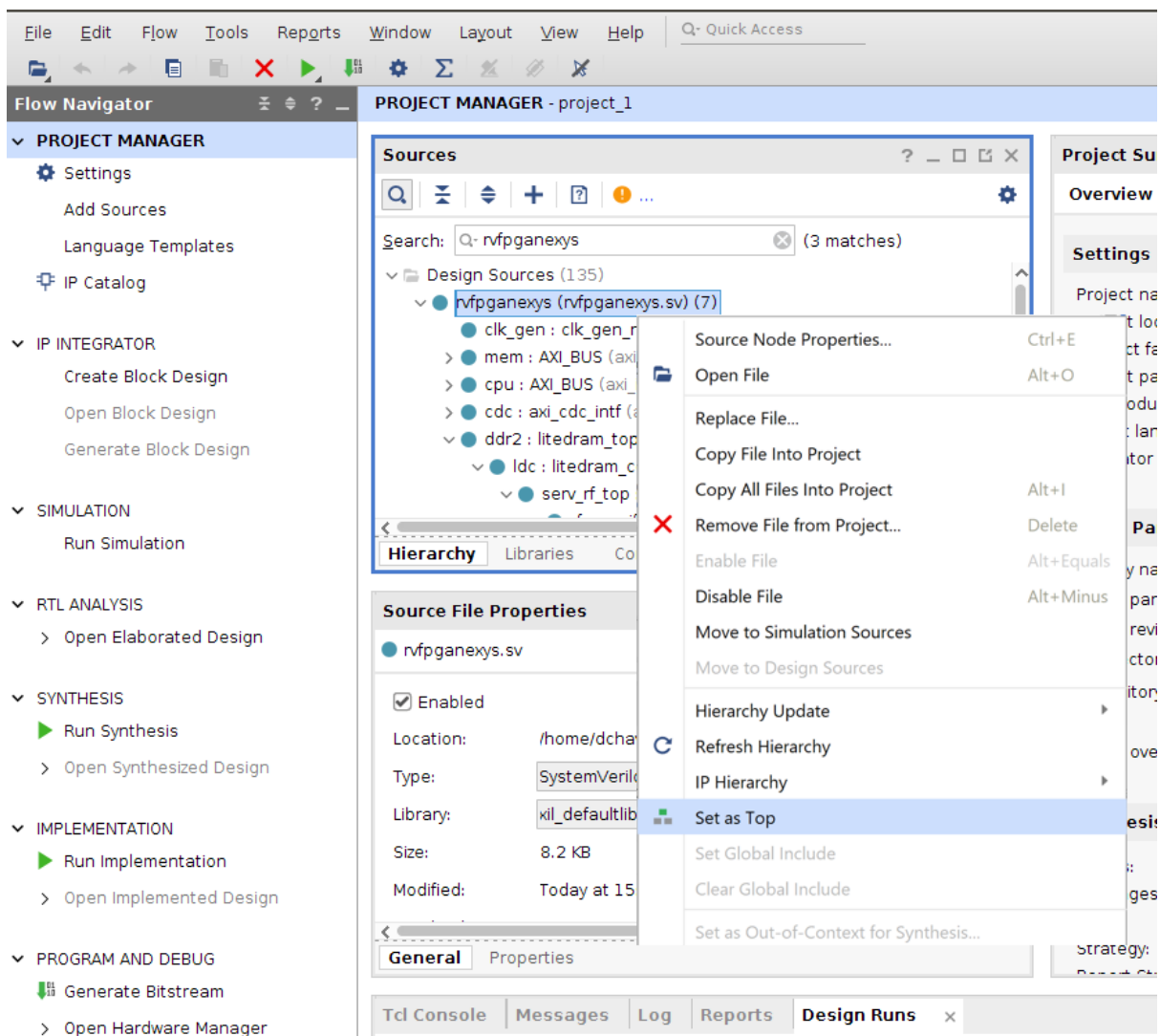


Figura 9. Definir rvfpganexys como módulo principal

Definir o ficheiro *common_defines.vh* como Global Include: Agora, ainda no painel *Sources*, em *Design Sources*, expanda o grupo de ficheiros *Non-modules* e clique em *common_defines.vh*. As propriedades do arquivo serão abertas no painel *Source File Properties*, logo abaixo do painel *Sources*. Clique em *Global Include* para marcar essa caixa (ver Figura 10). A hierarquia irá agora atualizar e incluir esse ficheiro em *Design*

Sources/Global Include. Observe que os ficheiros com erros de sintaxe desaparecerão na próxima etapa.

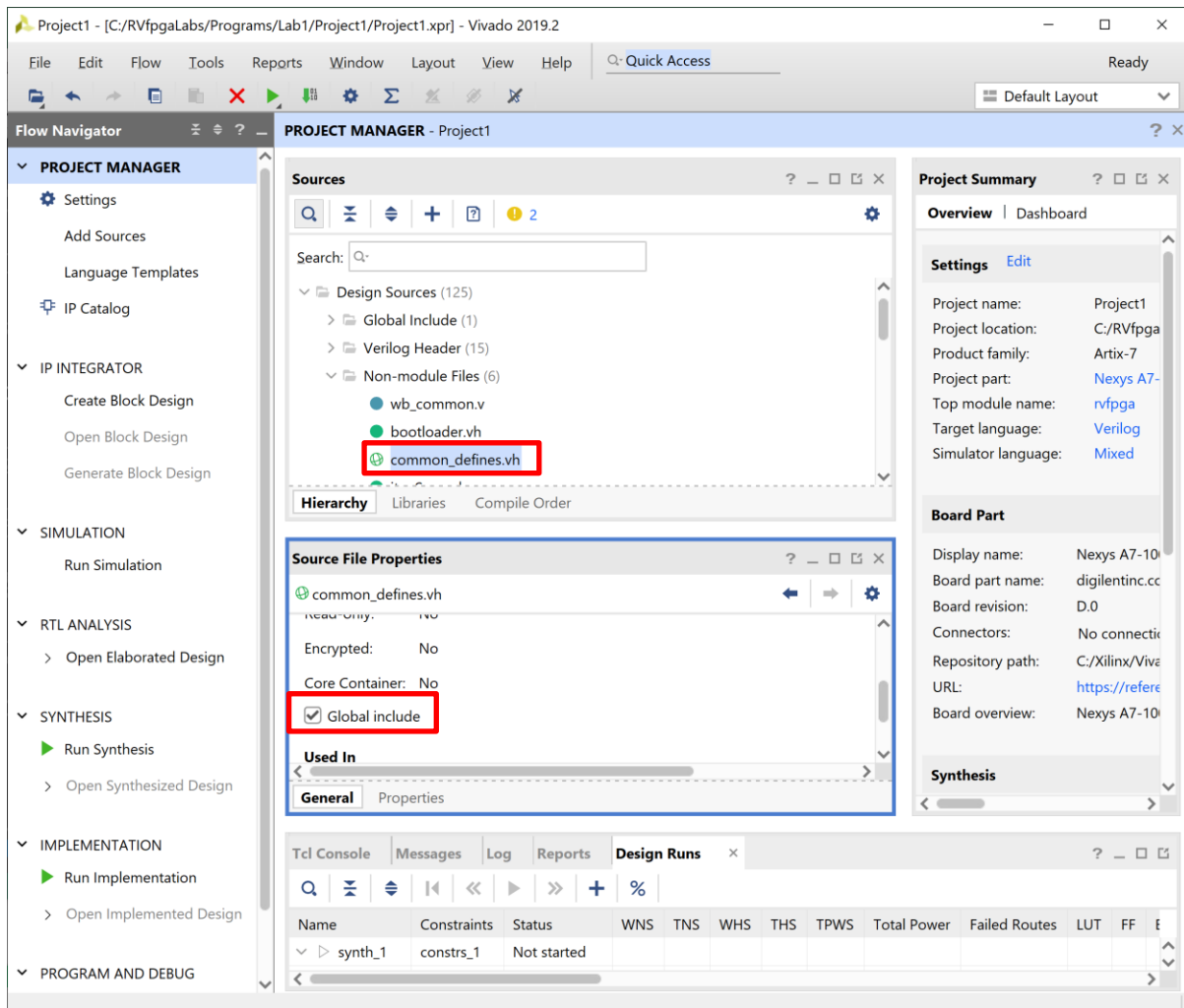


Figura 10. Incluir common_defines.vh como um ficheiro global

Adicionar boot_main.mem ao projeto: No painel *Flow Navigator*, clique em *Add Sources*, deixar a opção predefinida (*Add or create design sources*), e clique em *Add Files* (ver Figura 11). Vá para *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/BootROM/sw* e selecione *boot_main.mem* (como mostrado na Figura 11). A hierarquia irá atualizar e incluir esse ficheiro em *Design Sources/Memory File*.

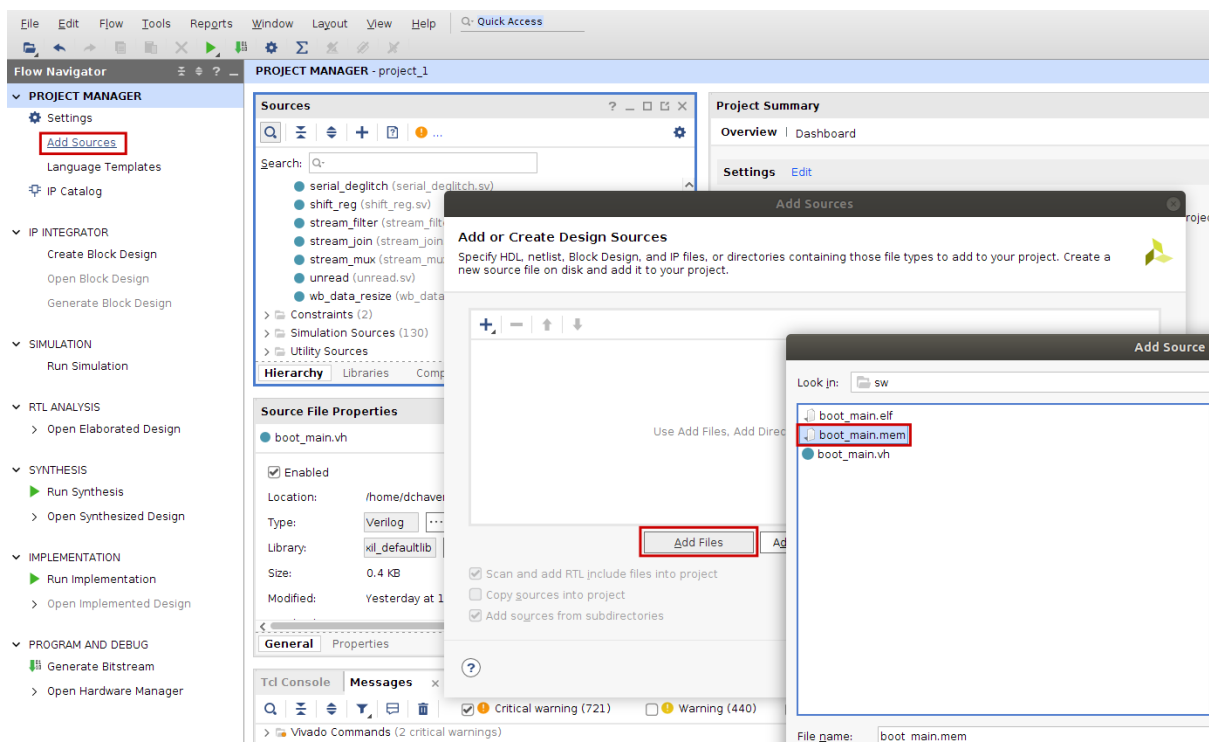




Figura 11. Adicionar ficheiro de memória boot_main.mem

O ficheiro (*boot_main.mem*) é utilizado para inicializar a ROM de inicialização do nosso SoC, invocando-o como parâmetro no ficheiro `[RVfpgaPath]/RVfpga/src/rvfpganexys.rv`:

```
25 module rvfpga
26     #(parameter bootrom_file = "boot_main.mem")
```

A secção 6.A do Guia de Introdução contém mais informações sobre este ficheiro.

Pastas a incluir: Finalmente, incluir duas pastas para a Plataforma Pulp (ver Figura 12). No painel *Flow Navigator* clique em *Settings*, e na janela que se abre clique em *General* e

depois em *Verilog options* (). Na nova janela, adicione os dois seguintes diretórios, clicando em  e navegando para os diretórios:

```
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/AxiInterconnect/pulp-platform.org__axi_0.25.0/include
[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/include
```

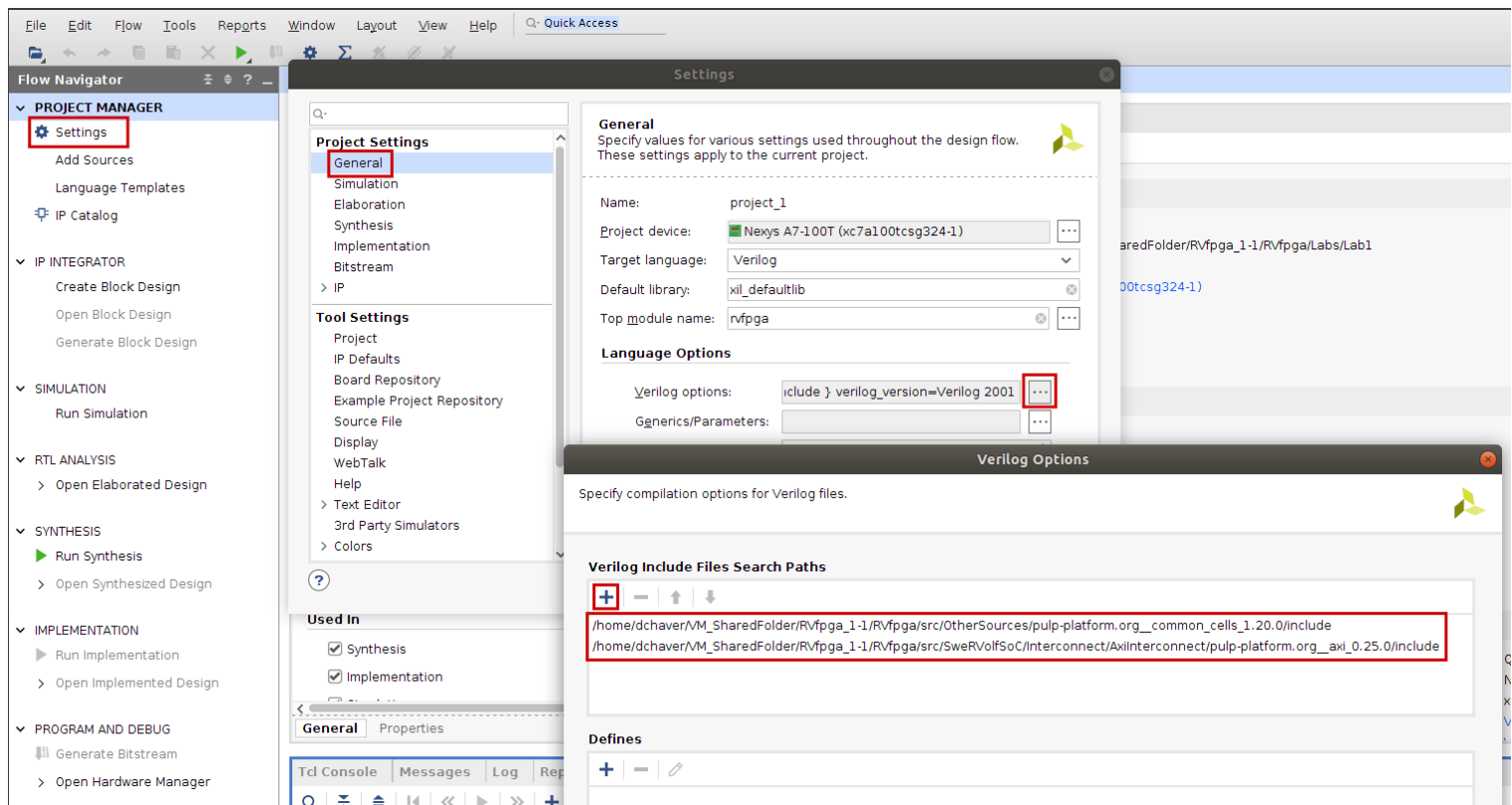


Figura 12. Pastas a incluir na Plataforma Pulp

Passo 6. Geração do ficheiro de configuração (Bitstream)

Agora clique em *Flow* → *Generate Bitstream* como mostra a Figura 13. Pode surgir uma janela que diz que não há resultados de implementação disponíveis e pede para lançar a síntese e implementação (ver Figura 14). Clique em Sim. Depois clicar OK na janela *Launch Runs* (ver Figura 15). Esta etapa sintetiza o RVfpgaNexys (como definido pelos ficheiros Verilog e SystemVerilog no projeto), mapeia-o para a FPGA, e cria o ficheiro de configuração (bitstream). Este processo demora normalmente 20-50 minutos, dependendo da velocidade do seu computador.

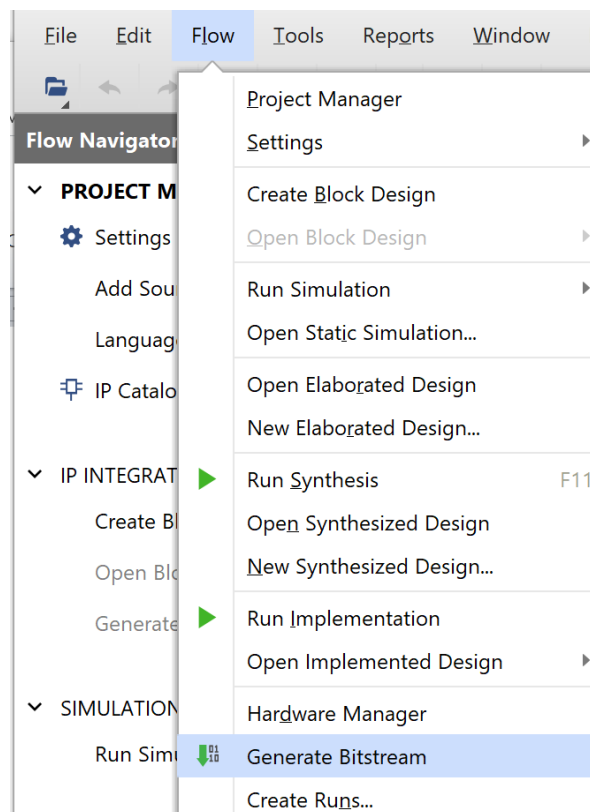


Figura 13. Geração do ficheiro de configuração (Bitstream)

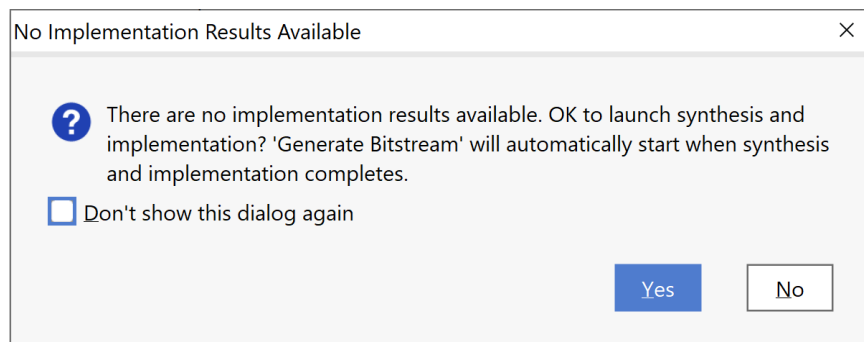


Figura 14. Janela de lançamento da síntese e implementação

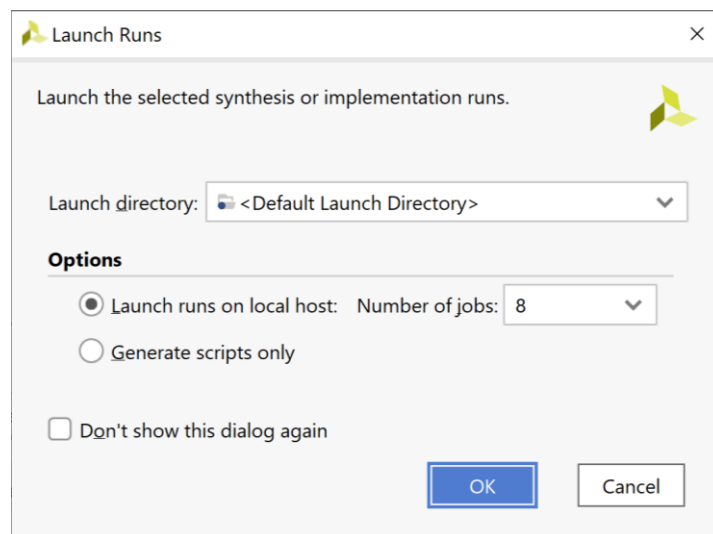


Figura 15. Lançamento das execuções

Após o bitstream ter sido gerado, surgirá uma janela como mostra a Figura 16. Clique no botão **X** no canto superior direito para fechar a janela.

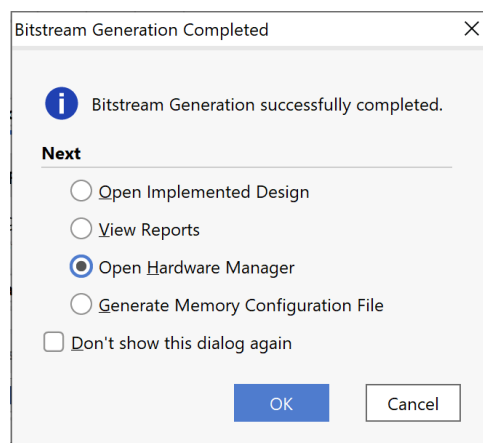


Figura 16. Geração de Bitstream Completa

Agora que construiu o sistema RVfpgaNexys, poderá reconstruir o RVfpgaNexys depois de ter feito modificações no Labs 6-10. Por agora, começará a utilizar o sistema RVfpgaNexys que acabou de construir para descarregar e executar programas nele usando PlatformIO.

IMPORTANTE: Pode encontrar o bitstream acabado de gerar pelo Vivado na pasta:

```
[RVfpgaPath]/RVfpga/Labs/Lab05/project_1/project_1.runs/impl_1
```

Recomenda-se que utilize o PlatformIO para descarregar o RVfpgaNexys para a placa Nexys A7. Este método foi descrito na Secção 6.A do Guia de Introdução ao RVfpga (GSG) em detalhe. Como também descrito no GSG para os diferentes exemplos (6.B a 6.H), depois de configurar o sistema RVfpgaNexys na FPGA da placa Nexys A7, irá utilizar o PlatformIO para descarregar e executar/depurar programas no RVfpgaNexys.

Também pode usar o Verilator, um simulador HDL, para simular programas em execução no RVfpgaSim, como descrito na Secção 7 do Guia de Introdução ao RVfpga. Estas simulações de nível RTL permitem visualizar sinais de hardware de baixo nível à medida

que o programa de software corre. Recorreremos muito ao Verilator nos Labs 6-10, à medida que amplia o Sistema RVfpga e testa e depura as suas alterações.