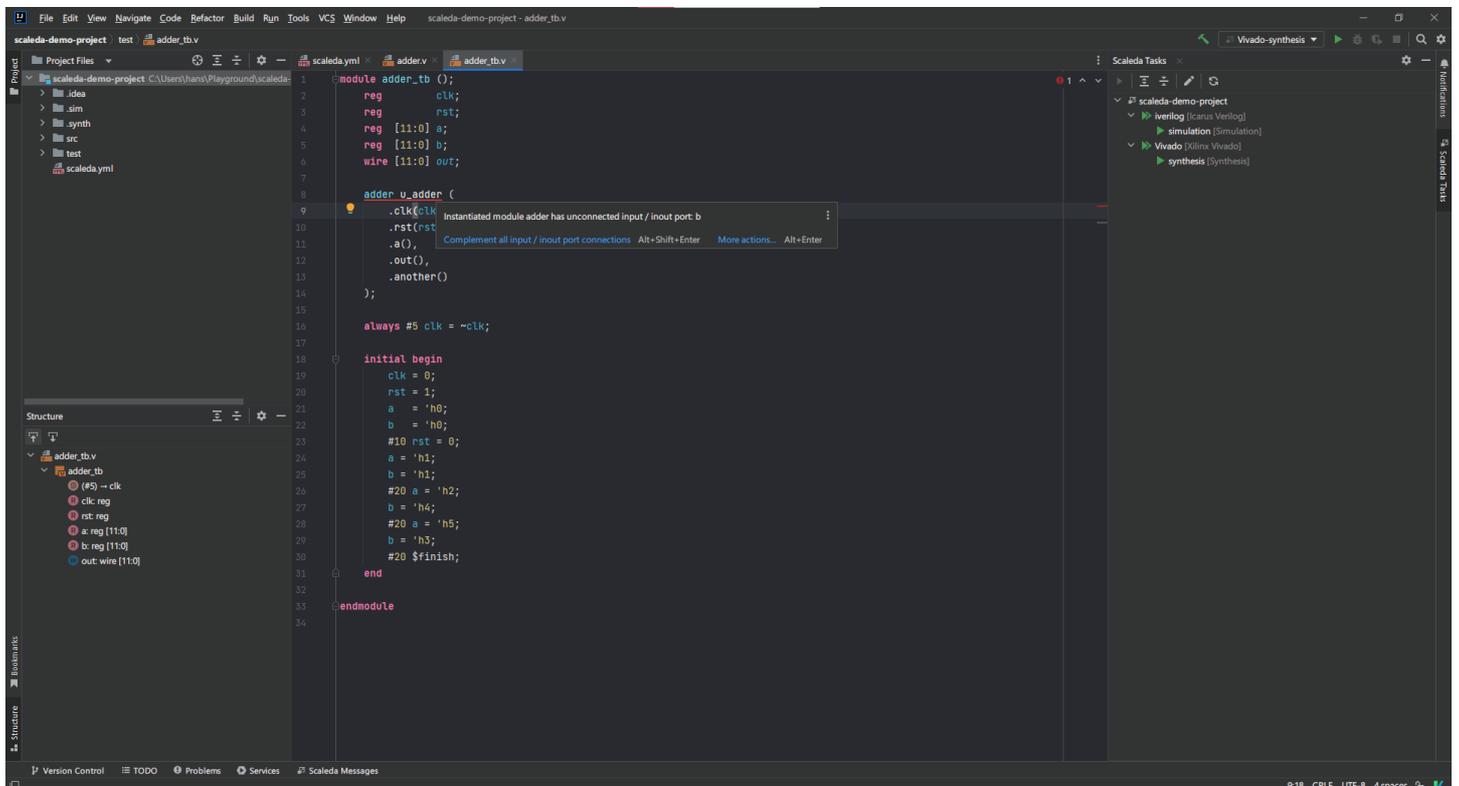


# Scaleda: 灵活、通用的 FPGA 开发平台

## 综述

Scaleda (=Scalable + EDA) 是一款用于 FPGA / Verilog 开发的集成开发环境。它致力于提供用户友好的开发平台，支持智能化的 Verilog 代码编写及便捷的仿真调试体验，并可接入 Vivado 等厂商工具链实现逻辑综合、实现等功能。

Scaleda 基于 IntelliJ 平台，是 [IntelliJ IDEA](#) 上的一款插件。用户可以在安装社区版 IDEA 后，安装 Scaleda 插件来使用它。简而言之，Scaleda 目前是 IDEA 上的一款集 Verilog 语法及语义提示、FPGA EDA 工具链管理与调用、FPGA 项目管理的插件。



Scaleda 的开发背景来源于我们对 FPGA 开发教学普及情况的观察。传统的 FPGA 开发围绕厂商提供的 EDA 软件进行。然而，这类 EDA 软件如 Vivado 通常比较笨重，并且实际开发体验较差。一些用户会选择使用 Visual Studio Code 并配合多个插件进行开发，但这种方式通常配置麻烦，对初学者并不友好，且实际体验参差不齐。基于对这些问题的考虑，我们希望设计一个灵活、通用的 FPGA 开发平台。Scaleda 的开发愿景便是提供：

- 简便而直观的配置，开箱即用的体验。在需要用户设置软件功能时，提供图形界面与详细介绍。
- 用户友好的开发体验。用户在编写代码时，应有实时的语法及语义检查，并能针对常见问题给予修正提示。
- 良好的可拓展性。项目配置等文件采用通用文本格式，便于合作开发，也便于软件移植。

Scaleda 是在 IntelliJ 平台上运行的。尽管 IntelliJ IDEA 是一个被设计用于 Java 开发的 IDE，但其强大的 API 扩展系统使得我们可以在其之上增加对其他语言的支持，这一点与 Visual Studio Code 是类似的。而相比之下，IDEA 更容易实现为「开箱即用」的开发环境，相比 Visual Studio Code 对用户来说配置更加简单，这也是我们选择它的原因。

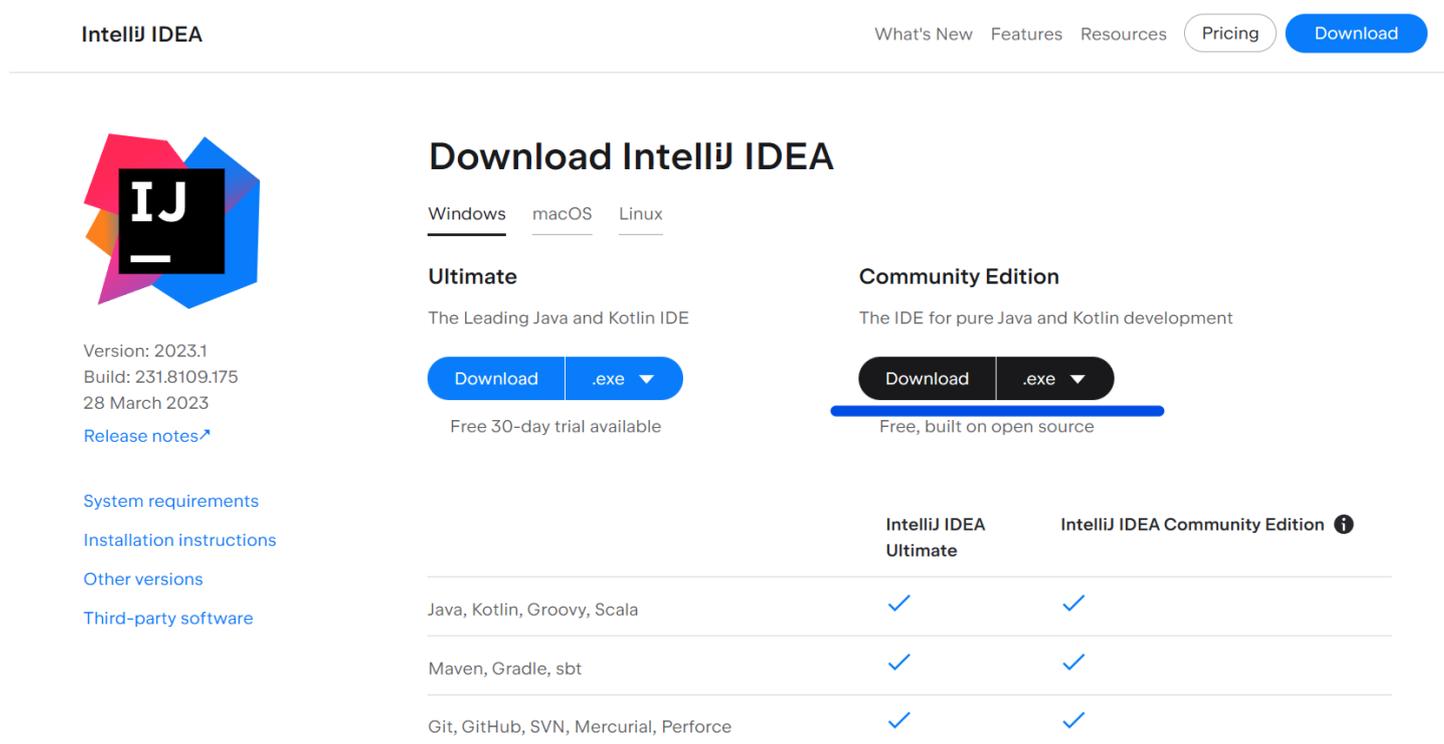
## 功能一览

Scaleda 目前有 2 种工作模式：Vivado 模式和独立运行模式。

- 您可以直接使用 Scaleda 打开现有的 Vivado 项目，编辑其中的源码，享受 Scaleda 带给您的开发体验，甚至调用 Vivado 中的综合、实现等任务。当然，您也可以同时打开 Vivado 在一旁，二者同时编辑一个项目。具体请见后文的《快速上手：Vivado 模式》一节。
- Scaleda 也有一套自己的项目系统。这套项目系统支持在同一个项目中指定不同的「任务」（如仿真、综合），每个任务可以有不同的顶层模块，可以在不同的平台（如 Xilinx Vivado）上执行，并提供了基本的调试功能。详见后文《灵活自如：Scaleda 项目系统》一节。

## 安装方式

Scaleda 是 IntelliJ IDEA 平台上的一款插件。需要 2023.1 及以上版本的 IntelliJ IDEA。可以在 <https://www.jetbrains.com/idea/download/> 下载免费、开源的社区版 IDEA（Community Edition）。



IntelliJ IDEA

What's New Features Resources Pricing Download

### Download IntelliJ IDEA

Windows macOS Linux

#### Ultimate

The Leading Java and Kotlin IDE

Download .exe

Free 30-day trial available

#### Community Edition

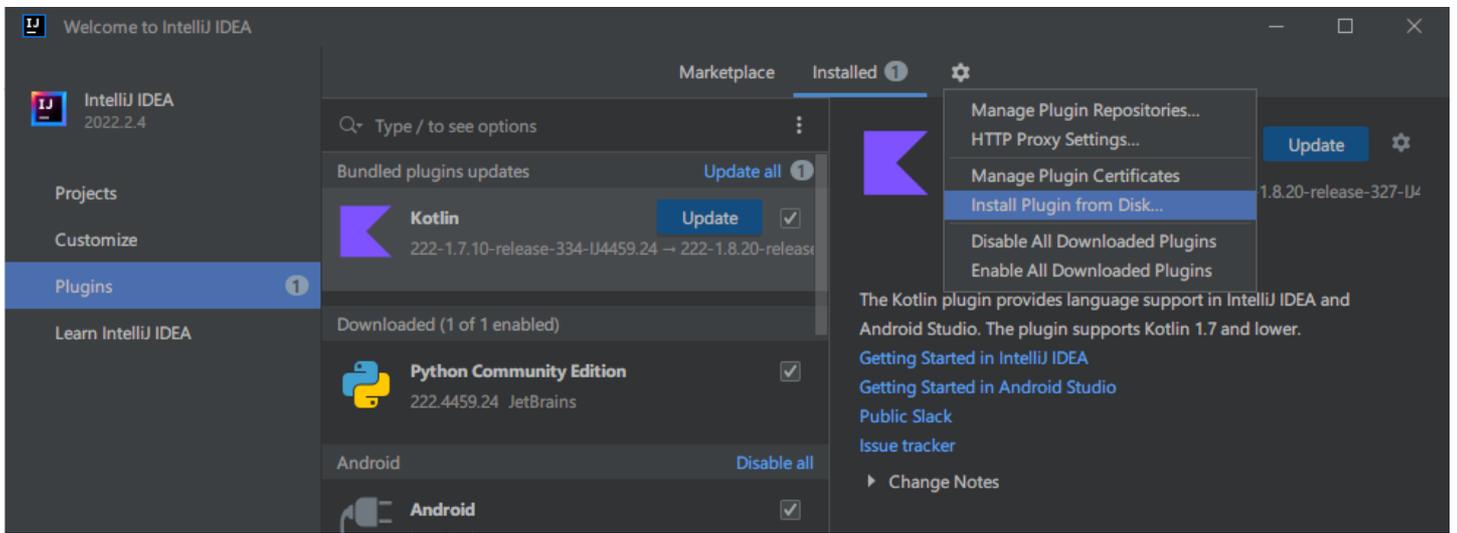
The IDE for pure Java and Kotlin development

Download .exe

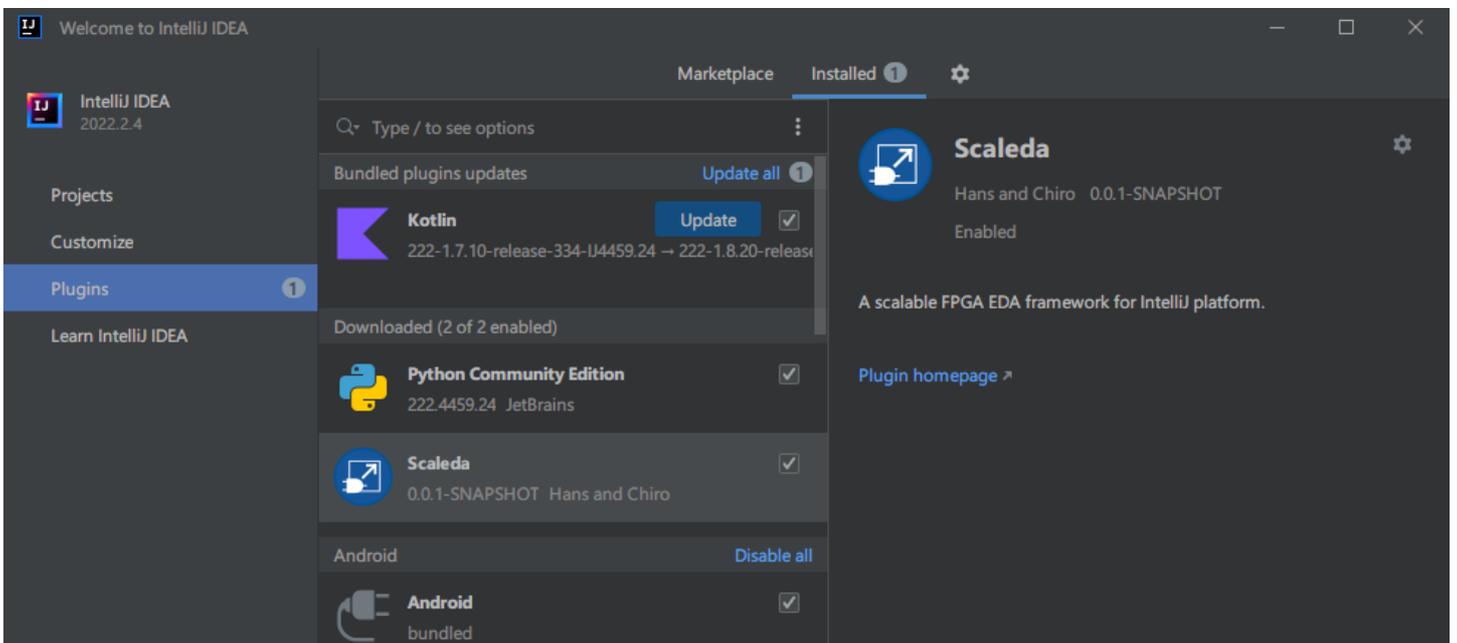
Free, built on open source

	IntelliJ IDEA Ultimate	IntelliJ IDEA Community Edition ⓘ
Java, Kotlin, Groovy, Scala	✓	✓
Maven, Gradle, sbt	✓	✓
Git, GitHub, SVN, Mercurial, Perforce	✓	✓

安装好 IDEA 后，在启动画面左方选择「Plugins / 插件」，然后点击右侧上方的齿轮按钮，选择「Install Plugin from Disk / 从磁盘安装插件」。



选择 `scaleda.jar` 文件，然后在弹出的警告窗口中，点击「Accept / 接受」。安装成功后，插件列表中将出现 Scaleda：



您可以开始体验 Scaleda 的功能了。阅读后文来了解 Scaleda 的使用。

## 快速上手：Vivado 模式

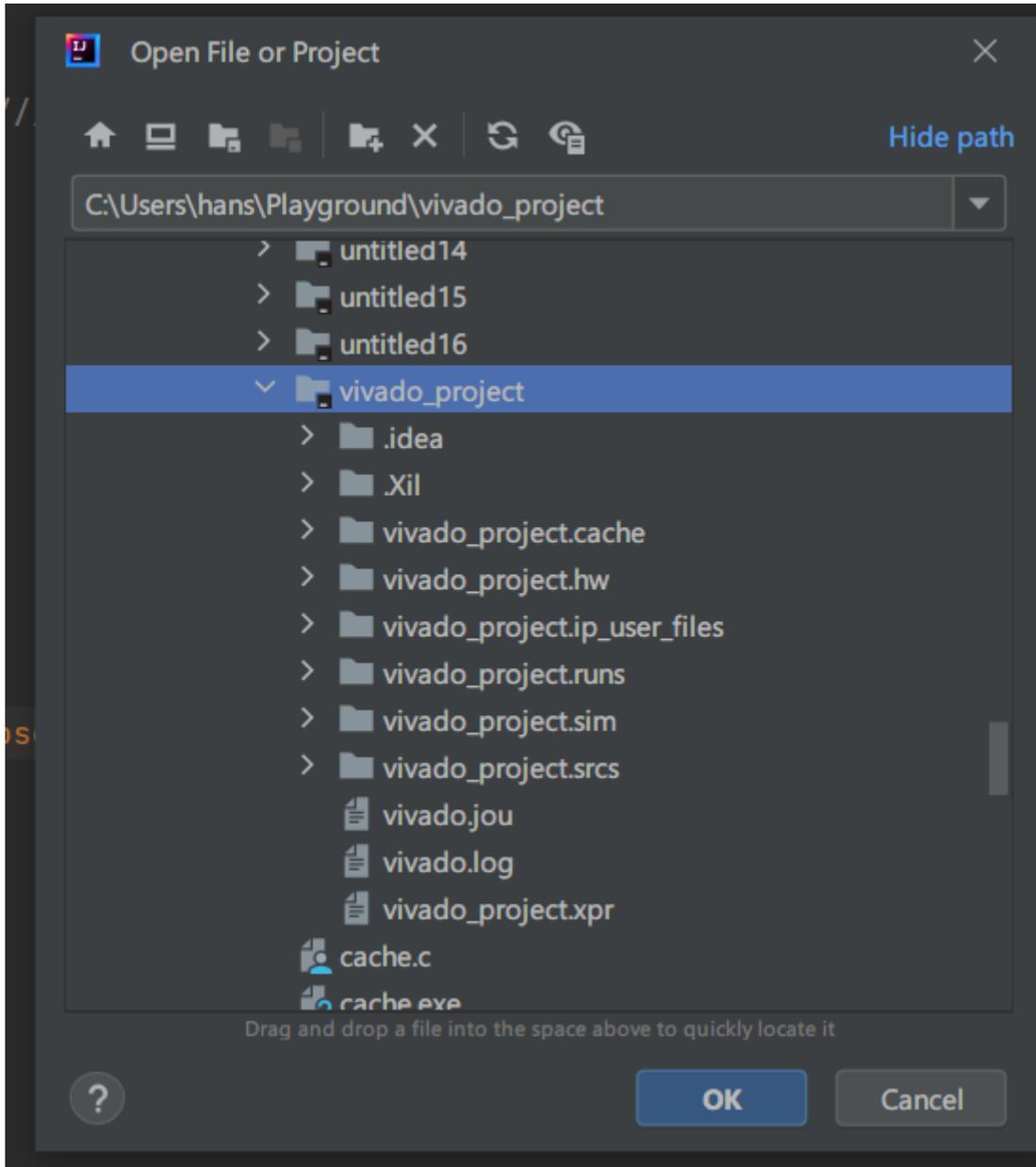
本节介绍 Scaleda 的 Vivado 模式。在 Vivado 模式下，您可以使用享受智能化的代码编辑，并在 Scaleda 中调用 Vivado 中的各项功能，如综合（Synthesis）、实现（Implementation）。部分功能如仿真（Simulation）等受限于 Vivado 本身，需要您手动切换到 Vivado 程序中执行。

文中，所有的「Scaleda」均指「安装了 Scaleda 插件的 IntelliJ IDEA」。

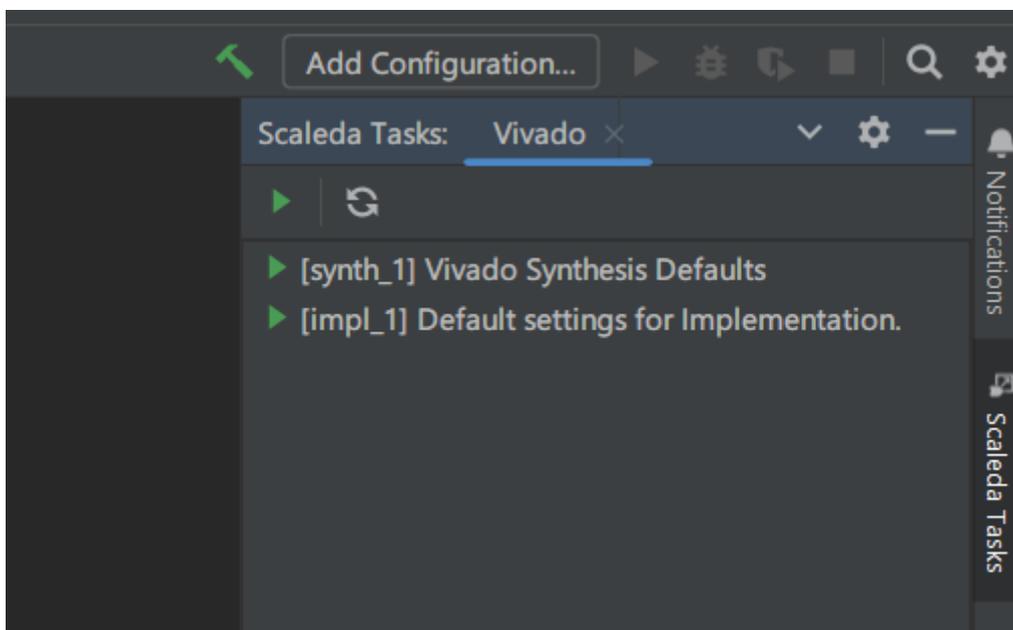
## 使用 Scaleda 打开 Vivado 项目

使用安装了 Scaleda 的 IDEA 打开 Vivado 项目所在的文件夹，Scaleda 会自动检测该文件夹下的 Vivado 项目文件（即 `.xpr` 文件）。

注意：打开时，选择 `.xpr` 文件所在的文件夹，不是选择 `.xpr` 文件本身。Scaleda 默认所有的 Vivado 项目都伴随一个文件夹。



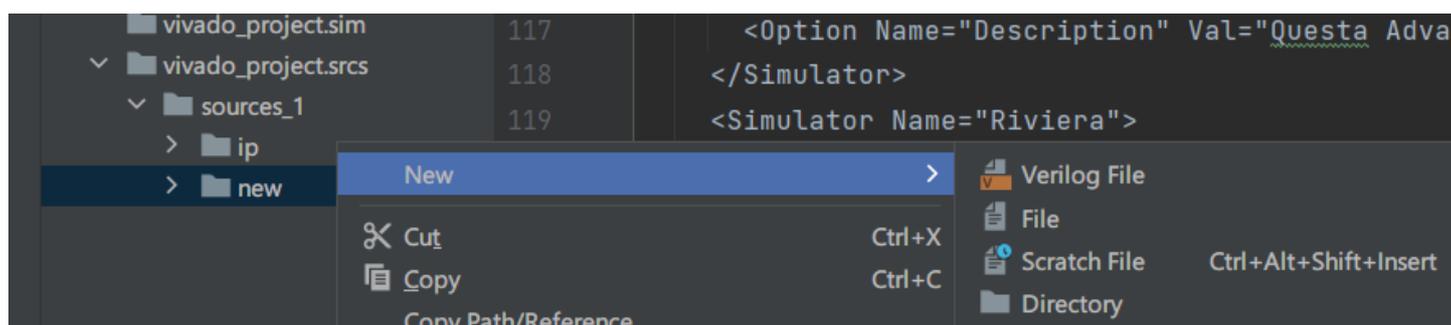
打开项目后，点击右侧的「Scaleda Tasks / Scaleda 任务」面板，在上方选择「Vivado」。若能加载出诸如「[synth\_1] Vivado Synthesis Defaults」的默认任务，说明成功打开了 Vivado 工程。



说明：若没有加载出这些默认任务，说明 Vivado 项目没有成功加载。您可以按一下此界面上的「Refresh Vivado / 刷新 Vivado 项目」按钮尝试重新加载。若仍然没有加载成功，请反馈此问题。

## 在 Vivado 项目中新建 Verilog 文件

请同时用 Vivado 打开这个项目，然后在 Vivado 中新建源码文件；也可以选择在 Scaleda 中，新建一个 Verilog 文件，然后再在 Vivado 中导入它。



注意 1：若您在 Vivado 中选择了添加源码文件，但添加的源码不在当前项目文件夹内，则可能造成问题。计划在后续更新中增加对外部代码的支持。

注意 2：受限于 Vivado 软件，目前不能自动让 Vivado 找到您在 Scaleda（即 Vivado 外部）新建的源码文件。

## 体验 Scaleda 编辑功能

得益于 IntelliJ IDEA 平台强大的拓展能力，Scaleda 实现了智能化的 Verilog 编辑器。本节将介绍您可以获得的开发体验。

### 语义级补全

```
1 module simple_adder(  
2     i  
3     input  
4     inout  
5     Press Enter to insert, Tab to replace Next Tip
```

## 信号提示

```
1 module simple_adder(  
2     input clk,  
3     input [11:0] a,  
4     input [11:0] b,  
5     output reg [11:0] c  
6 );  
7 always @(posedge clk) begin  
8     c <= a + b;  
9 end  
10  
11 endmodule
```

代码中所有的信号会根据它们的类型使用不同的字体表示。例如，上图中，所有紫色斜体表示「线网」类型，而紫色正体表示「寄存器 (reg)」类型。您可以将鼠标移动到某个信号上，浮窗会展示此信号的定义。您也可以按住 Ctrl 点击以跳转到那个信号的定义处。

同样的提示也应用于模块的实例化上。您可以轻松地通过浮窗查看某个被实例化的模块的原始模块头（即定义了模块参数和模块端口的部分），跳转到模块定义，或是查看某一个端口和参数的属性。

```
13 module top ();
14     reg clk, a, b;
15     wire c;
16     simple_adder u_simple_adder (
17         .clk(clk)
18         .a (a),
19         .b (b),
20         .c (c)
21     );
22 endmodule
23
```

```
module simple_adder (
    input clk,
    input [11:0] a,
    input [11:0] b,
    output reg [11:0] c
)
```

vivado\_project

## 实时问题检测

Scaleda 会实时检测您的代码，检查其中的错误和可能造成问题的设计。

## 语法成分出错与缺失

Scaleda 会检查您的代码是否存在语法问题，并标记出存在语法问题的位置。

```
7     always @(posedge clk) begin
8         c <= a + b;
9     end
10
11 endmodule
12
13 module top ();
```

```
missing ';' at 'end'
```

```
input [11:0] b
```

vivado\_project

## 驱动语句检查

Scaleda 也会进行一些基本的语义检查。例如，当您尝试驱动一个 `input` 端口、尝试在 `always` 结构中驱动一个 `wire` 信号，或是使用 `assign` 给一个 `reg` 赋值，都会出现对应的错误提示。

```
5     output reg [11:0] c
6 );
7     assign c = 'h12;
8     always @(p
9         c <= a
10 end
```

```
Can only assign to net (wire, tri, etc.) or 'output' port with keyword 'assign'
```

## 模块实例化检查

Scaleda 会检查模块实例化时的连线情况和参数赋值情况。如果存在缺失的连线，Scaleda 会给出提示，并可自动补全连线。

```
14 module top ();
15     reg clk, a, b;
16     wire c;
17     simple_adder u_simple_adder (
18         .clk(clk),
19         .b (b),
20         .c (c)
21     );
22 endmodule
23
```

Unconnected port found: a

Complement port connections (with named connection) Alt+Shift+Enter More actions... Alt+Enter

注意：此自动补全会自动使用「指名连接」方式（即 `.foo (bar)` 的方式）为您建立所有的连接语句，但您仍然需要手动去将顶层模块的信号接入。

结合此功能，您可以方便地完成一个大型模块的实例化。

### always 逻辑块问题检测

Scaleda 可以检测出与 `always` 逻辑块有关的一些潜在问题。例如：

- 一般来说，对于边沿敏感的 `always` 块，其建议使用非阻塞赋值即 `<=`；而对于电平敏感的 `always` 块，则建议使用阻塞赋值即 `=`。如果在代码中混用，则会出现提示。

```
21 always @(*) begin
22     a <= 0;
23 end
24 endmodule
```

Ambiguous assignment in combination logic always block

- 对于电平敏感的 `always` 块，其敏感信号若未在块内被显式使用，则会被作为时钟处理，这可能在综合时产生多时钟问题。因此，若您的 `always` 敏感信号有未被使用的，则会出现提示。

```
22 always @(posedge clk or posedge rst) begin
23     // if (rst) b <= 0;
24     b <= a;
25 end
```

Ambiguous clock in event control

- 如果一个信号在多个 `always` 块中被驱动，则有很大概率会在综合时产生问题（多驱动问题）。Scaleda 会标记出这样的信号，并可在多处驱动之间来回跳转。

```
22     always @(posedge clk or posedge rst) begin
23         if (rst) b <= 0;
24         b <= a;
25     end
26
27     always @(posedge clk) begin
28         b <= 0;
29     end
```

Net b is multi-driven in another always construct

Navigate to multi-driven Alt+Shift+Enter More actions... Alt+Enter

## 潜在的锁存器检测

对于组合逻辑的 `if` 或 `case` 语句，若没有形成完整的逻辑通路（即 `if` 总是与 `else` 配对、`case` 总是有 `default`），则会在综合中产生锁存器，这是不好的设计。Scaleda 会对这一问题进行标注。

```
27     always @* begin
28         if (a) b = 0;
29     end
30 endmodule
```

Latch detected in conditional statement

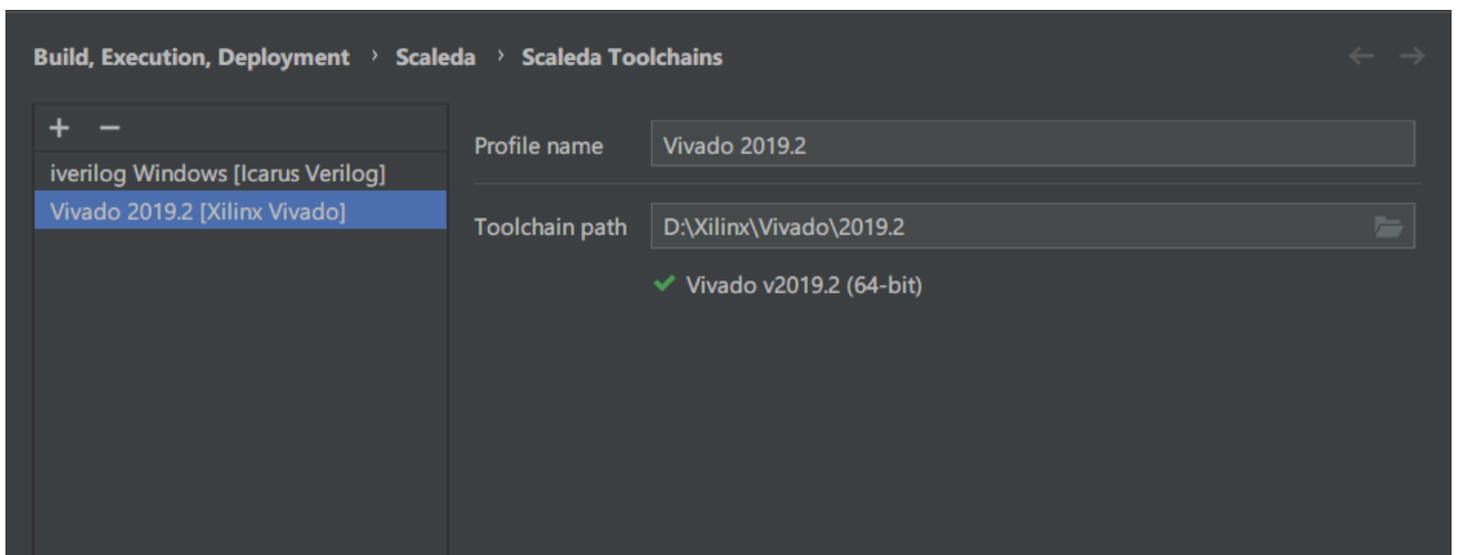
## 代码格式化

Scaleda 能自动控制代码中的缩进。例如，当您在合适的位置键入一句 `begin` 并按下 Enter，可补全出整个 `begin - end` 结构，并调整新代码的缩进。

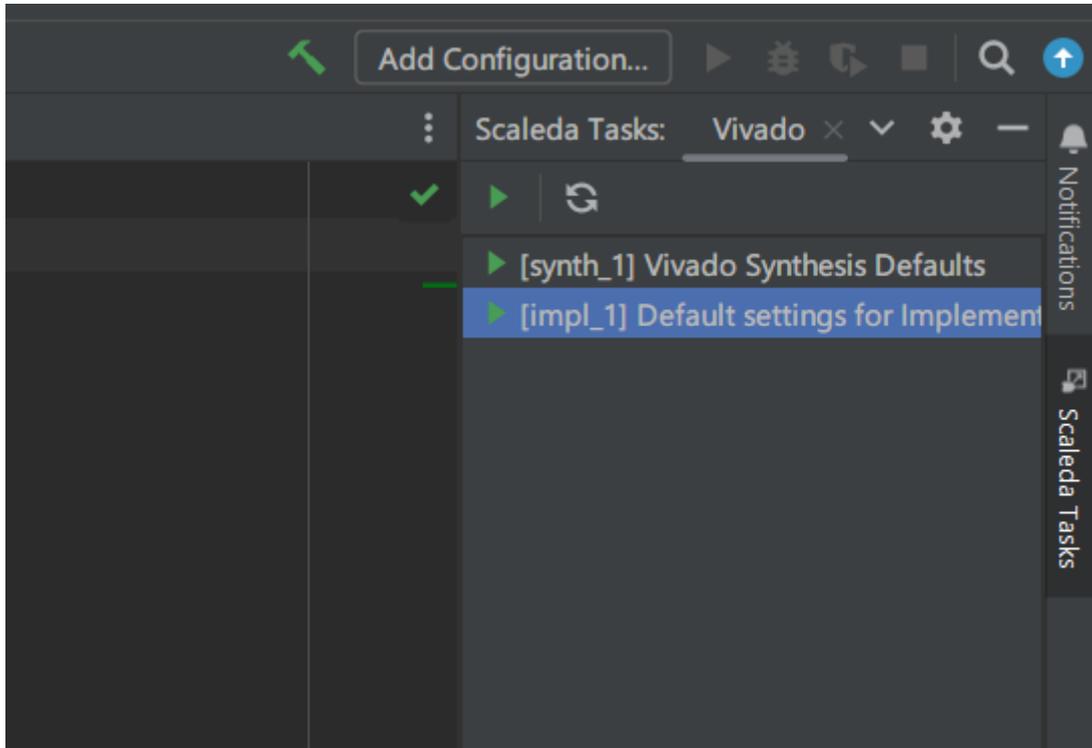
Scaleda 也支持代码全文格式化和部分格式化。选择 Code | Reformat Code 或按 Ctrl + Alt + L 可以格式化当前文件。注意，含有语法错误的文件无法被格式化。

## 执行 Vivado 综合及实现

您可以在 Scaleda 中调用 Vivado 的综合及实现等功能。在那之前，您首先需要配置本地 Vivado 工具链的安装位置。选择 File | Settings | Build, Execution, Deployment | Scaleda | Scaleda Toolchains，添加一个 Vivado 类型的配置，然后保存。

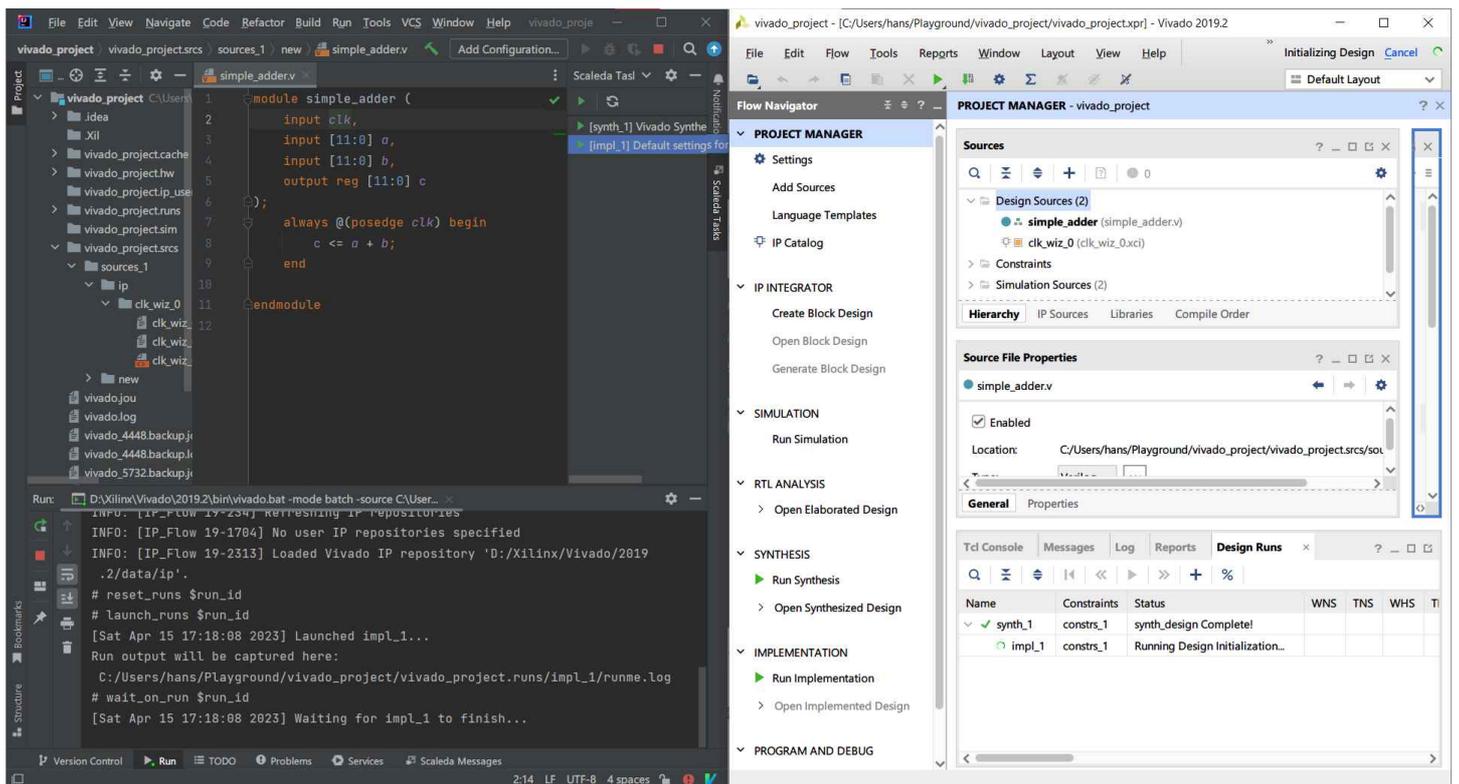


配置好 Vivado 工具链后，您可以打开右侧的 Scaleda Tasks 面板，在上方选择「Vivado」，然后双击综合或实现任务来执行。



注意：一个新建的 Vivado 项目，默认存在综合任务 `synth_1` 和实现任务 `impl_1`。它们对应 Vivado 的 Synthesis 和 Implementation。

您可以同时使用 Vivado 打开这个项目。Scaleda 中的调用会与 Vivado 中的状态同步。如下图。



仿真需要在 Vivado 中手动启动。这是 Vivado 的设计限制。

## 愿景与反馈

Vivado 模式是本项目重点支持的一个方向，它旨在提供一个辅助 Vivado 进行 FPGA 相关教学的工具，既解决了 Vivado 编辑体验差的问题，又相比 Visual Studio Code 的配置更加友好，并通过语法与语义提示帮助使用者养成良好的开发习惯。

目前 Vivado 模式仍然存在的问题有：

- 受限于 Vivado 软件逻辑，建议您在 Vivado 中新建源码后，再切换到 Scaleda 中编辑代码，并且不要在 Vivado 中添加不在项目目录之下的代码文件。在 Scaleda 中新建的代码不会被 Vivado 识别，必须在 Vivado 中手动导入。将考虑使用一些策略解决这一问题。
- Vivado 默认不会给新项目增加仿真任务，您需要手动切换到 Vivado 中仿真。这是 Vivado 的设计，可能不会修正。
- 您在 Vivado 中新建 IP 核后，必须完成一次 IP 核综合，否则无法索引到 IP 核及相关端口。

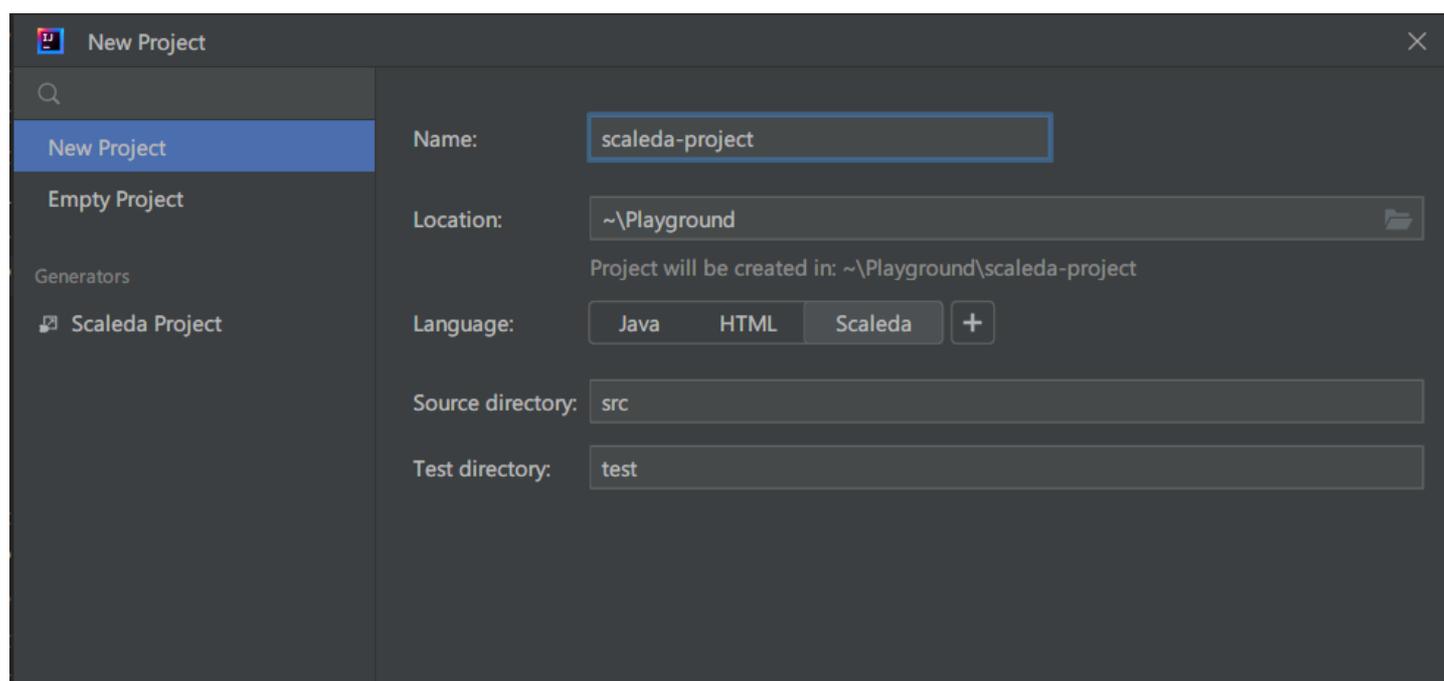
对于 Vivado 模式的其他使用问题，包括建议及 bug，都请向我们反馈。

## 灵活自如：Scaleda 项目系统

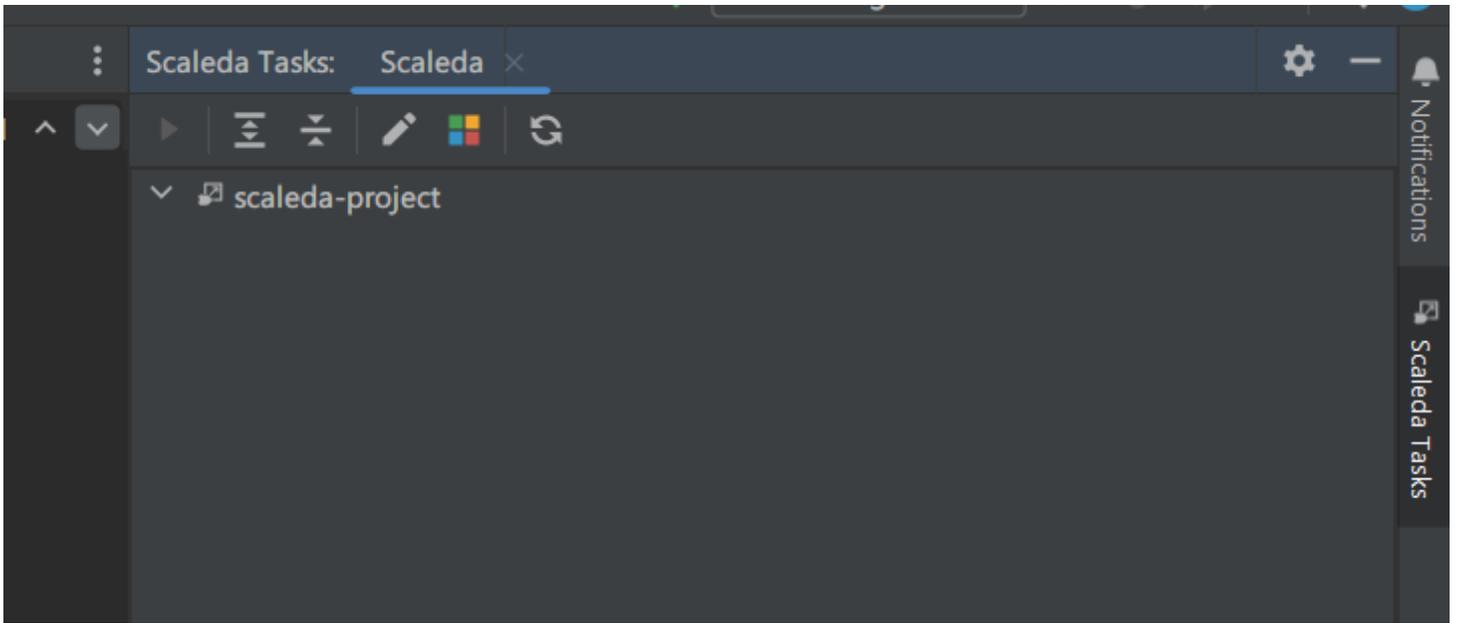
本节介绍 Scaleda 自己设计的项目系统。借助这套项目系统，您不仅可以实现在一个项目中同时管理多个顶层模块，还可以对一个项目应用多种不同「工具链」，也可以在同一工具链下针对不同平台（例如，不同的芯片型号）执行互相独立的任务。

### 新建一个 Scaleda 项目

您可以在 Scaleda 中选择 File | New | Project，然后在 Language 一栏选择 Scaleda，设置项目名称及项目路径后，即可完成新建。



当您打开一个 Scaleda 项目后，可在软件右面板 Scaleda Tasks 上方选择 Scaleda。您应该看到一个树形结构：



注意：若项目没有成功加载，您可以点击此面板上方的刷新按钮。

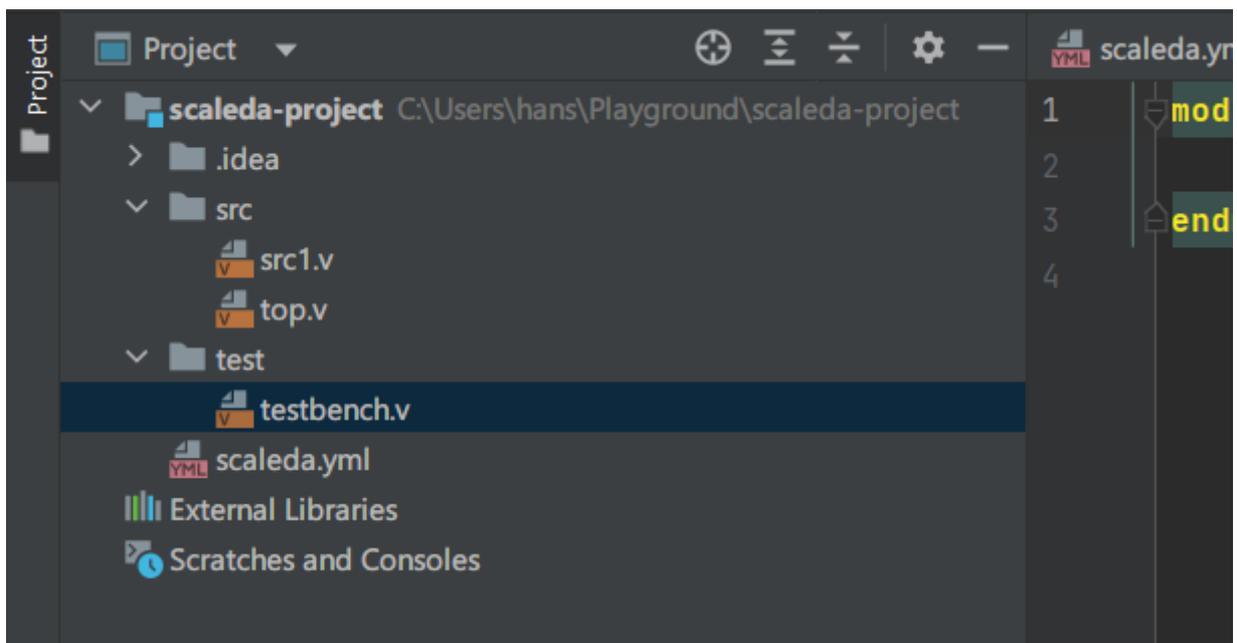
## 项目、目标平台与任务

此处介绍 Scaleda 的项目结构。

### HDL 文件

HDL 文件是项目的核心。在一个 Scaleda 项目中，HDL 文件（目前仅支持 Verilog）分为两类——源码和测试文件。源码指普通的 Verilog 模块，而测试文件对应 testbench。一个项目可以有多个源码文件，也可以有多个测试文件。

默认情况下，源码放在项目下 `src` 文件夹中，而测试文件在 `test` 文件夹里。



### 目标平台

一个「目标平台」 (target) 指一个可以用来完成 EDA 功能的平台。一个目标平台需要选定工具链的类型，如 Vivado 或 Icarus Verilog。根据工具链类型的不同，目标平台还可能需要设置额外参数，如对于 Vivado 平台，就需要设置器件型号。

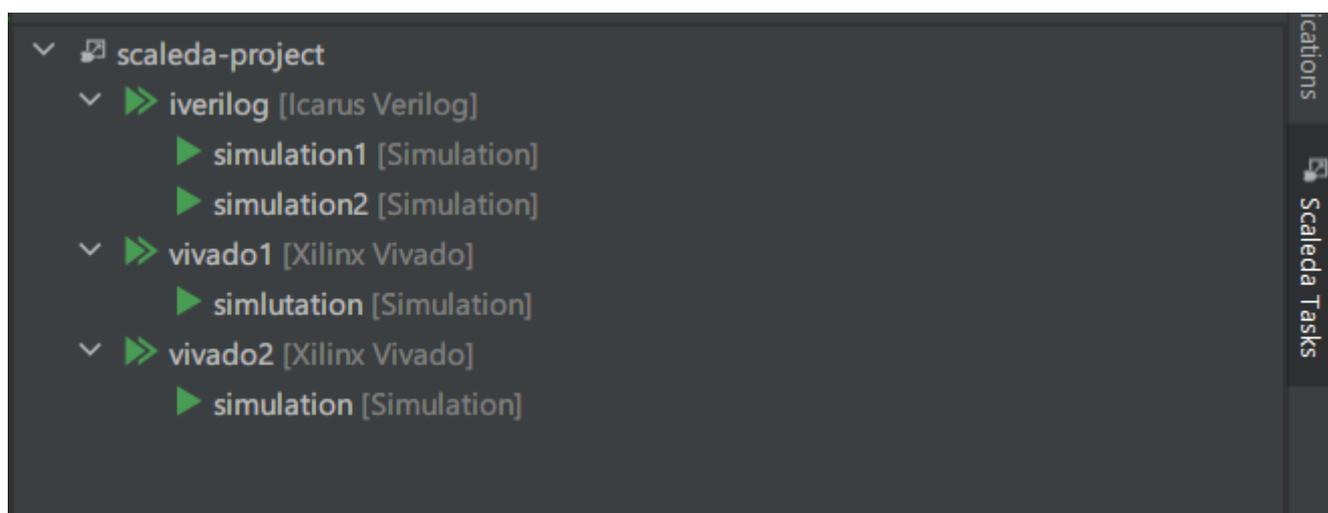
一个项目可以有多个目标平台，包括多个使用同一工具链的目标平台。

## 任务

一个「任务」 (task) 指在一个目标平台上的可以被执行的事件。在 Scaleda 中，任务有如下四种类型：

- 仿真 (simulation)，执行此任务会使用指定的 testbench 模块运行行为仿真，并展示波形图供调试。
- 综合 (synthesis)，执行此任务会使用指定的顶层模块进行逻辑综合，生成电路对应的 EDIF 网表文件。
- 实现 (implementation)，执行此任务会使用输入的 EDIF 网表文件进行实现，生成对应器件的比特流文件。
- 编程 (programming)，执行此任务会将输入的比特流文件编程到 FPGA 硬件上。

Scaleda Tasks 面板中会以树形结构，展示当前项目中已有的任务：



上图中，项目中有 3 个目标平台——使用 Icarus Verilog 的平台 iverilog，和使用 Xilinx Vivado 的平台 vivado1 与 vivado2。其中，iverilog 平台下有 2 个仿真任务，它们互相独立，可以设置不同的顶层模块。而 vivado1 平台和 vivado2 下各有 1 个仿真任务。vivado1 与 vivado2 可以使用不同的器件型号，也可以有不同的约束文件集合。

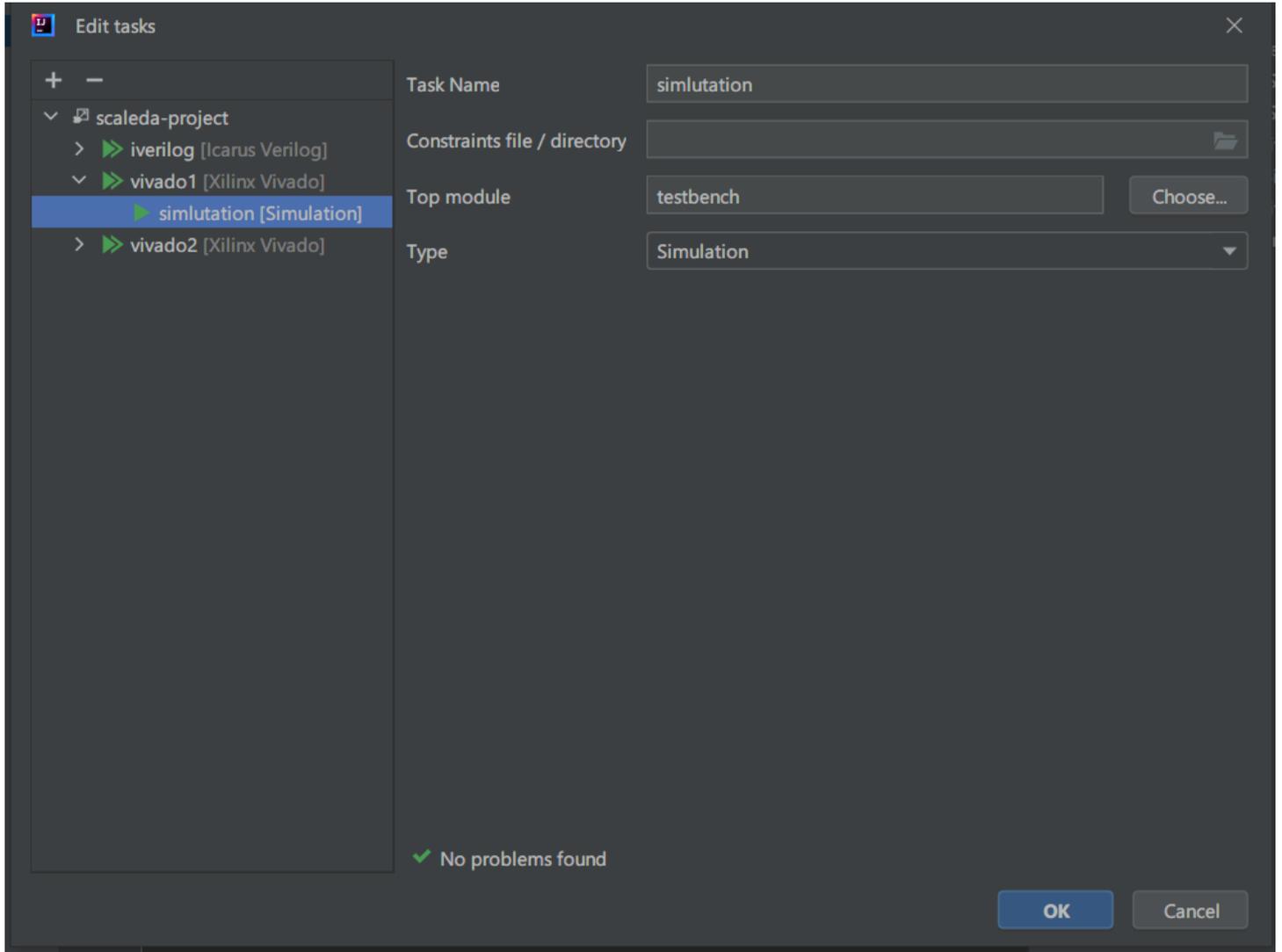
## 创建 Verilog 文件

您可以在左侧项目文件树中，找到您的源码文件夹（一般是 `src`）和测试代码文件夹（一般是 `test`），右键选择 New | Verilog File，即可新建 Verilog 代码。

关于编辑 Verilog 代码的细节，请参见前文 Vivado 模式中已经有的介绍。

## 编辑项目的目标平台和任务

点击右面板 Scaleda Tasks 上方的铅笔按钮，即可编辑项目中的目标平台等信息。



- 界面左侧的树型结构，展示着当前项目中的各个目标平台及每个平台下的任务。您可以在左侧进行点选，右方将加载您所点选的项目。您也可以点击上方的加号或减号按钮来新增或删除任务。
- 界面右侧是对应的编辑界面。
  - 当您在左侧点选的是项目根（如上图的 scaleda-project）时，右侧可以设置的项目包括：
    - 项目名称
    - 项目的代码目录及项目的测试代码目录。请谨慎更改这两个项目。
    - 项目全局顶层模块（可选）。如果您在此处设置了顶层模块，那么项目中其他没有指明顶层模块的任务，都会自动回滚到此处的设定。
  - 当您在左侧点选的是目标平台时，右侧可以设置的项目包括：
    - 平台名称与平台类型。
    - 平台的顶层模块（可选）。如果您在此处设置了顶层模块，那么此平台之下的所有任务，在没有指明顶层模块时，都会自动回滚到此处的设定。

- 平台的额外设置，如器件型号。
- 当您在左侧点选的是任务时，右侧可以设置的项目包括：
  - 任务名称
  - 任务类型（仿真、综合、实现、编程）
  - 顶层模块。如果您在此任务的上级设置过顶层模块，那么此处可以不填。

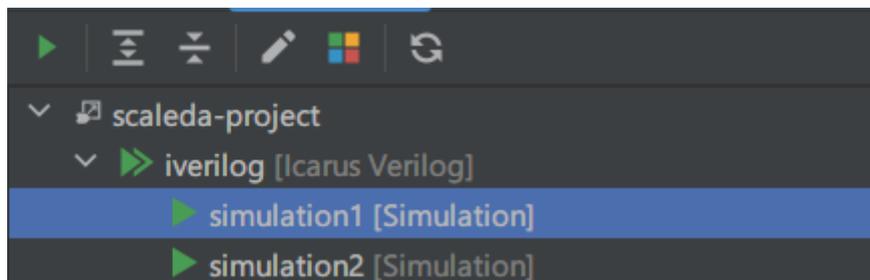
## 各平台功能介绍

在使用对应平台之前，您需要在设置中，添加该工具链的安装位置——例如，在使用 Icarus Verilog 之前，您需要设定本机上安装 iverilog 的位置。参见前文中关于设置 Vivado 安装位置的说明。

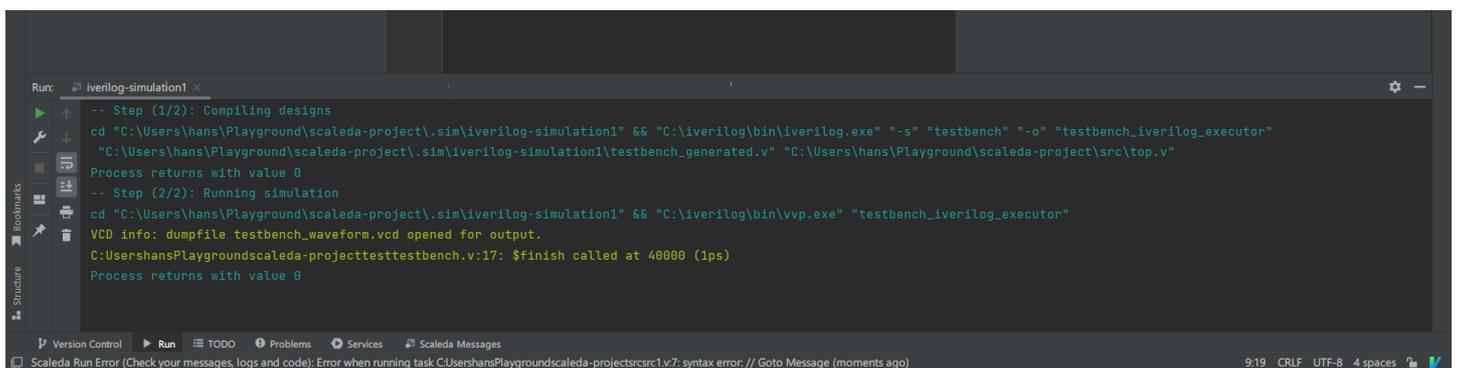
## Icarus Verilog (iverilog)

Icarus Verilog 是一个开源的 Verilog 仿真器。它体积小、速度快，对于简单的设计，完全可以替代商业仿真器进行仿真操作。Scaleda 支持使用 Icarus Verilog 对设计进行仿真。

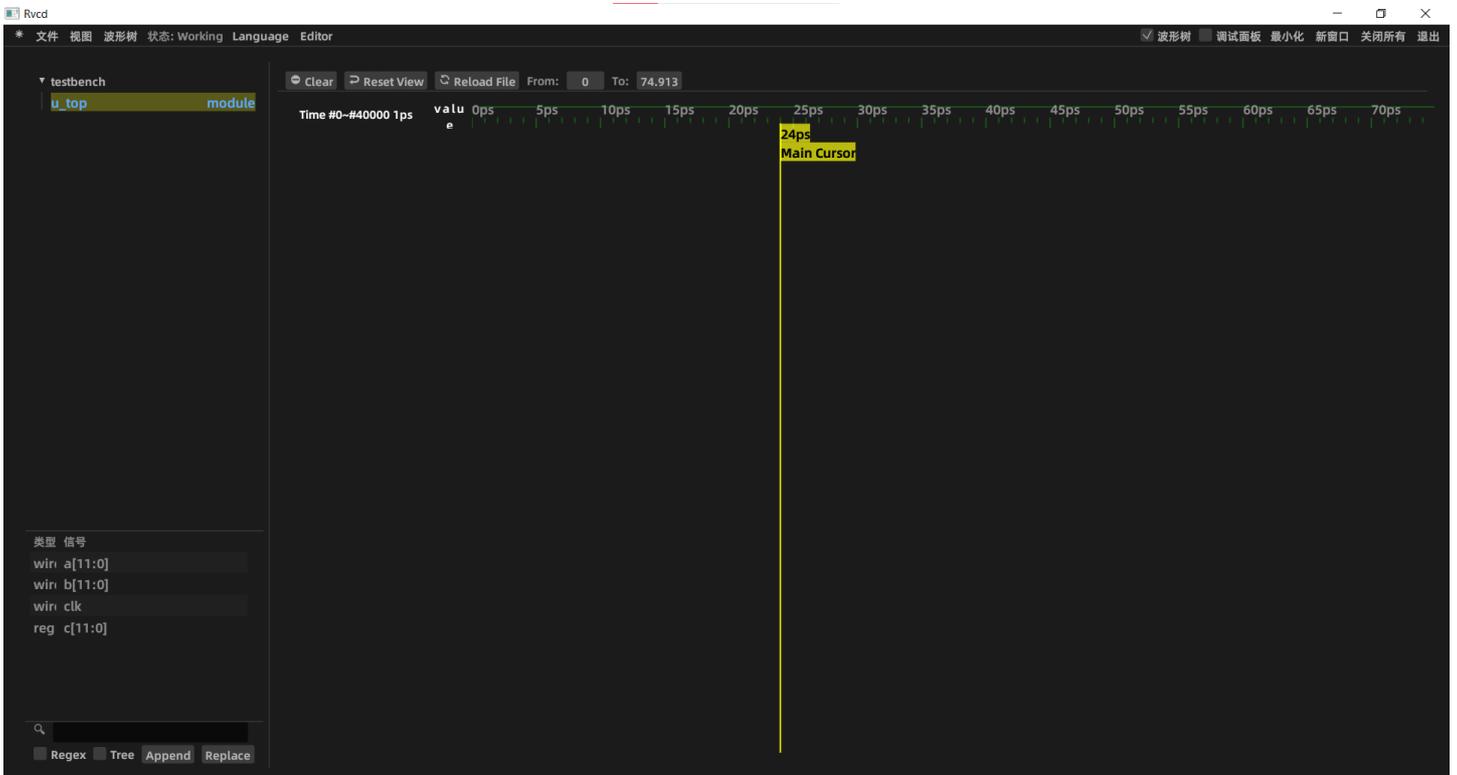
在 Scaleda 项目中，新建一个目标平台，工具链类型选择「Icarus Verilog」，并在该平台下新建一个「Simulation | 仿真」型任务。如图：



在右侧边栏中双击您想要执行的仿真任务，将自动启动仿真过程。仿真输出将展示在下方面板。



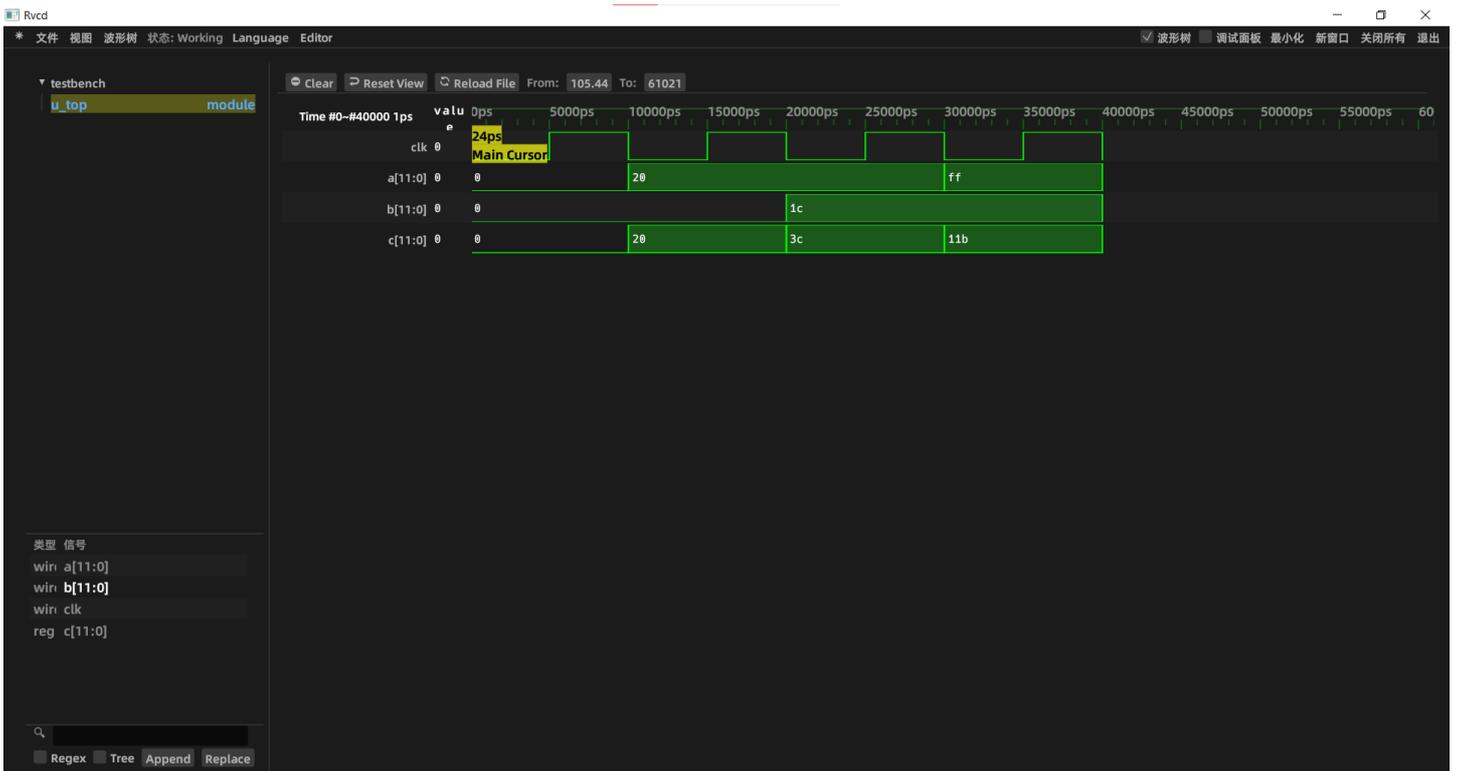
当仿真成功结束时，会自动打开波形查看器。



## 波形查看器的使用

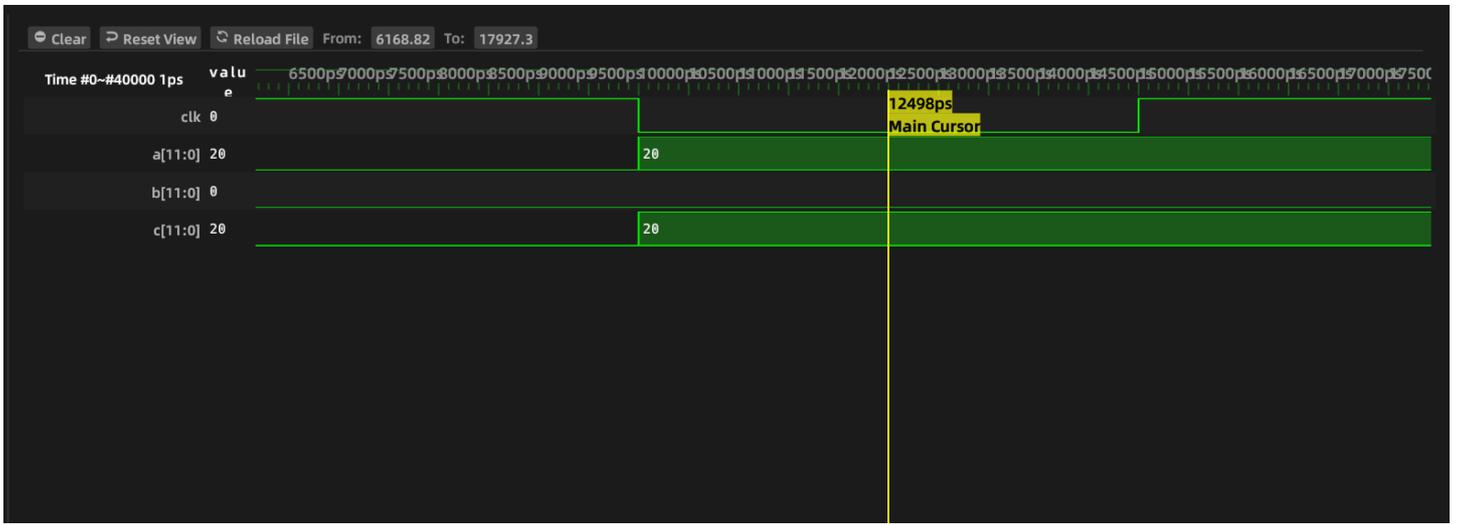
本小节介绍波形查看器的使用。上图中，左上方展示的是从测试顶层模块（即 testbench）开始的模块层级关系，左下方展示的是上方选中模块中的信号，而右侧是波形展示区。

例如，我们需要查看 testbench 模块中实例化的 `u_top` 模块内的信号，则在左上角点击 `u_top`。这时，左下角就会展示出 `u_top` 内的信号。双击信号即可将其加入波形面板。

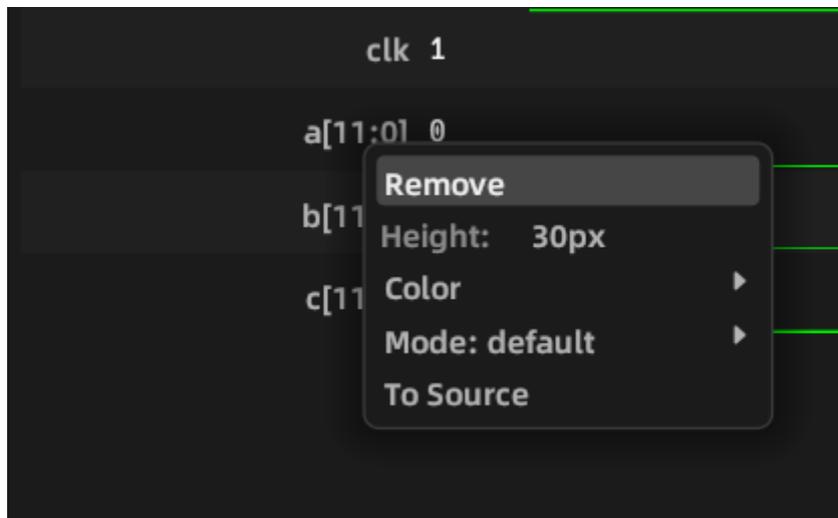


也可以直接双击左上方的模块，会直接将整个模块中的所有信号加入面板。

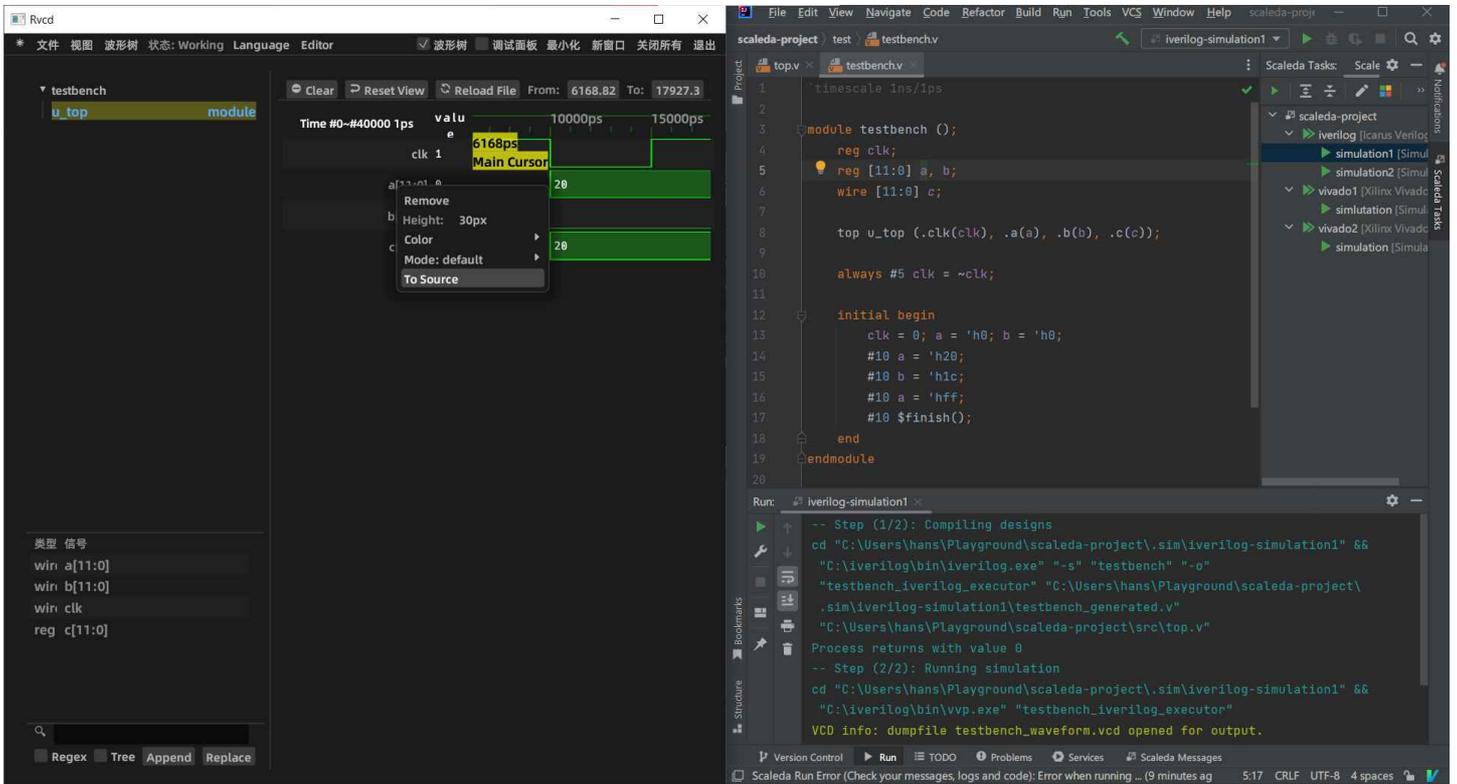
在面板中，按住 Ctrl 并滚动鼠标滚轮进行缩放；按住 Shift 并滚动鼠标滚轮可以前后移动。



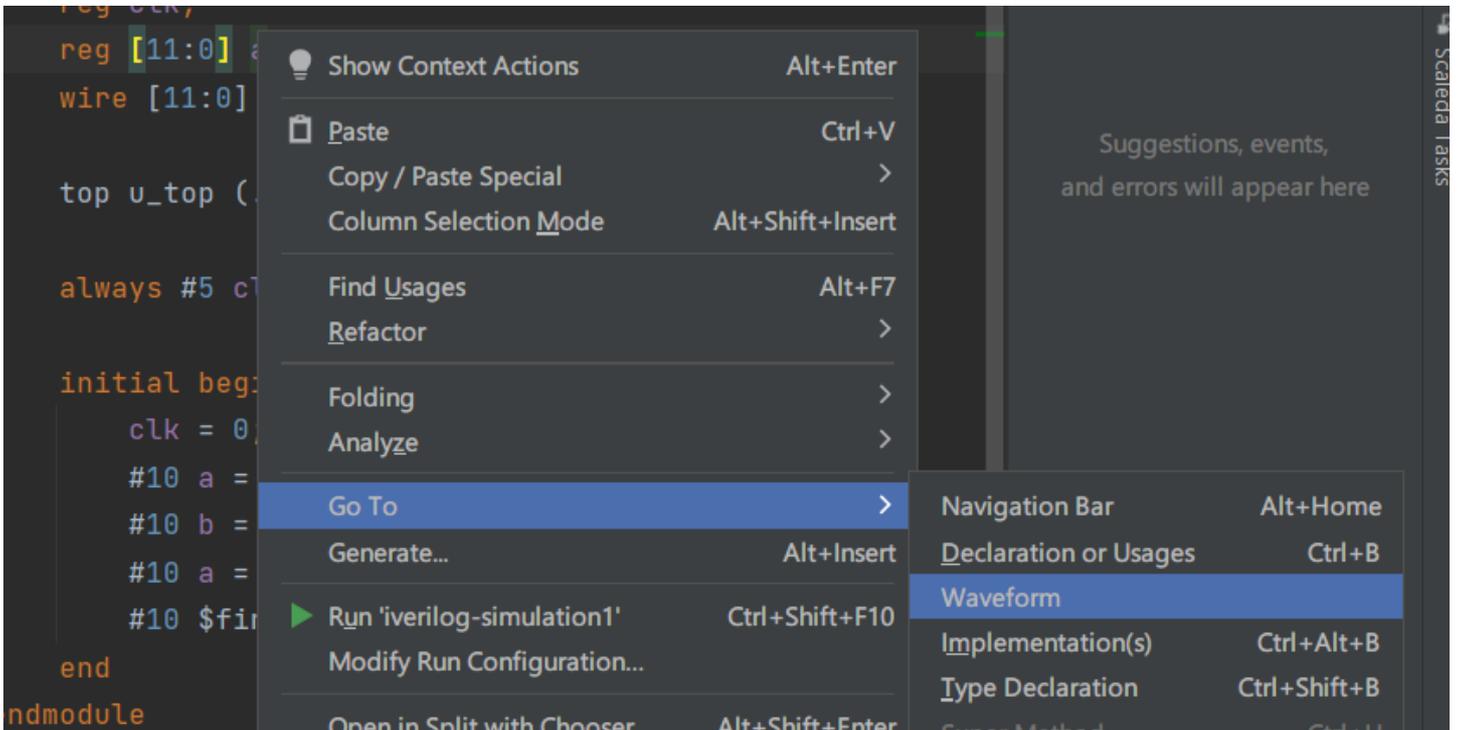
您可以在波形面板中，右键某个信号的名称，移除这个信号、更改显示颜色，或以模拟量形式显示它。



您还可以右击这个信号，选择「To Source / 到源码」，选择目标模块，即可跳转到源码中的对应信号。



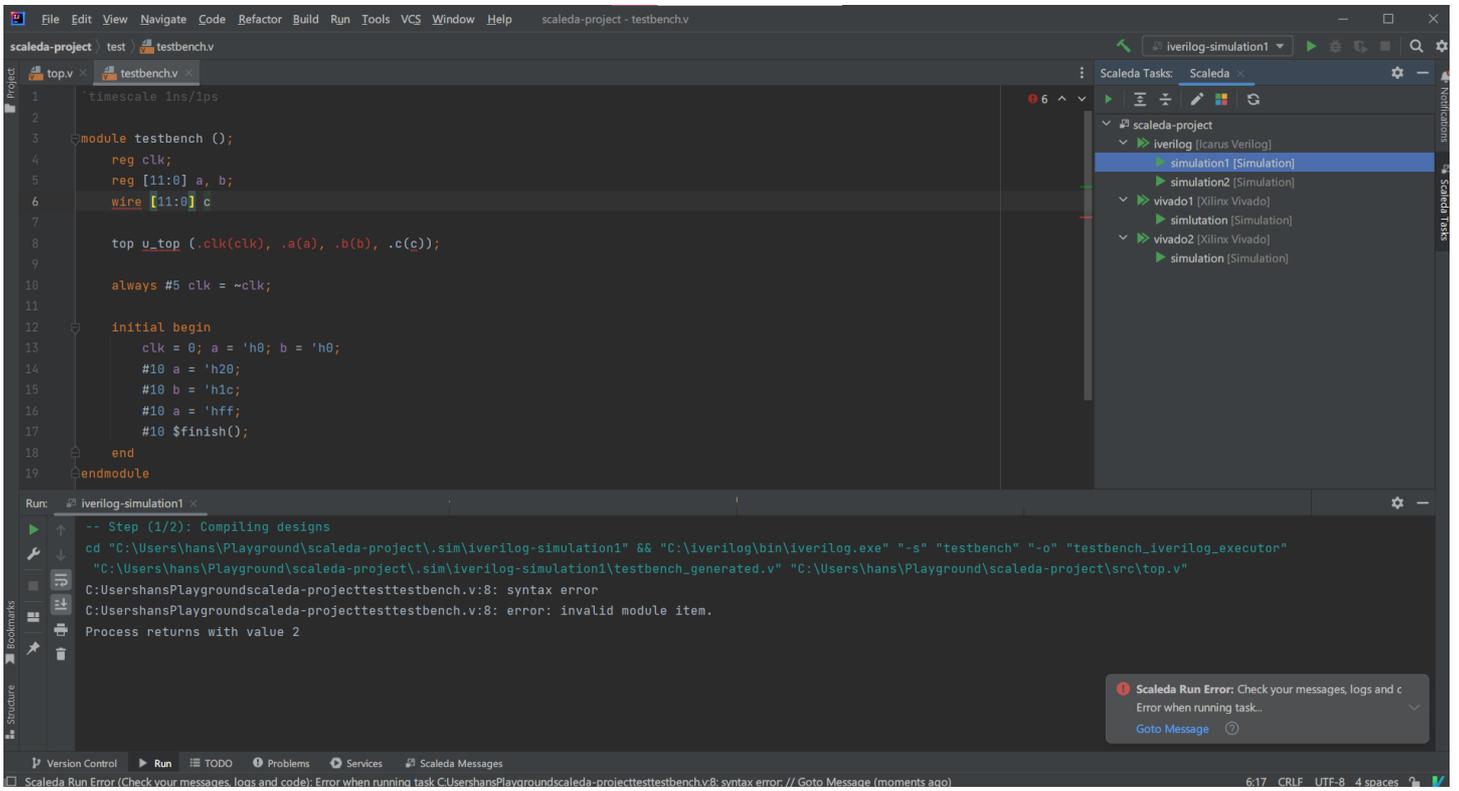
同样地，您在源码中右击某个信号，选择 Go To | Waveform，可以在波形查看器中添加它。



注意：目前受限于系统限制，您需要手动在两个窗口中切换。

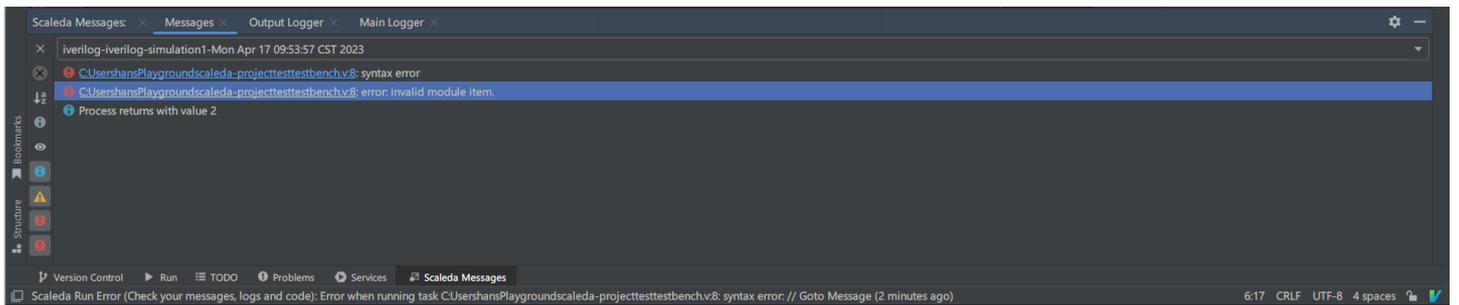
## 错误提示与处理

如果您的代码存在问题，仿真会失败。您可以在下方运行面板中看到原始的 iverilog 输出。下图是删除了代码中一个分号 ; 后运行的效果。



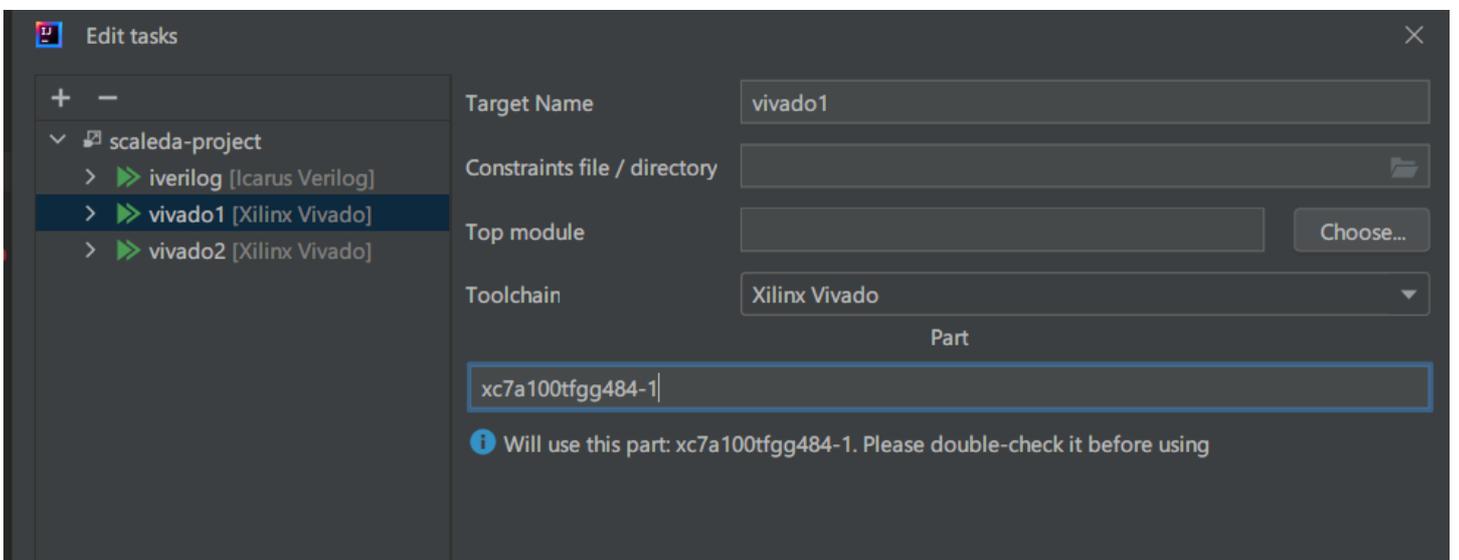
注意：上图中，运行面板内的文件路径的斜杠消失了，这是显示 bug。

您可以在下方找到 Scaleda Messages 面板。该面板会图形化显示所有的输出。



## Vivado

Scaleda 支持在 Scaleda 项目中使用 Vivado 工具链。在 Scaleda 项目中新建一个目标平台，选择工具链类型为「Xilinx Vivado」。需要在新建时提供完整、有效的器件型号，如 `xc7a100tffgg484-1`。

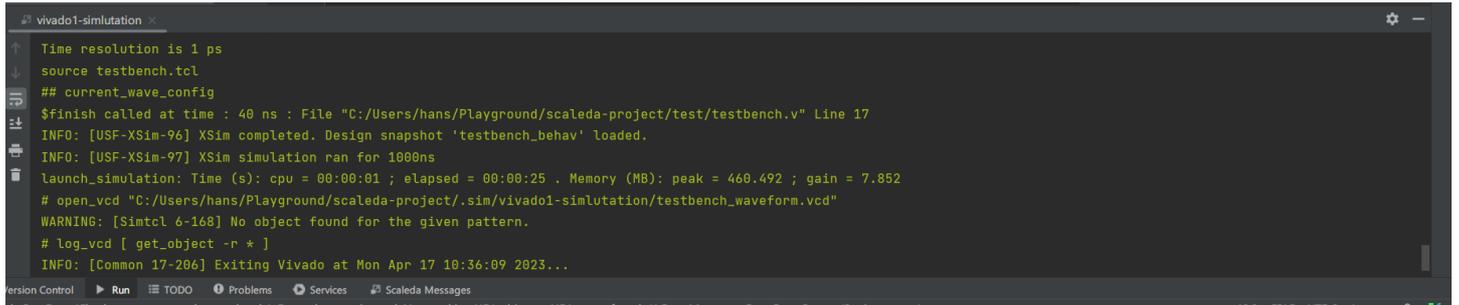


随后，您可以在此目标平台下，新建「仿真」「综合」「实现」及「编程」（指将比特流写入芯片）四种类型的任务。

## 使用 Vivado 仿真

您可以使用 Vivado 作为仿真器使用。与 iverilog 类似，您需要在 Vivado 类型的目标平台下，新建一个「Simulation / 仿真」类型的任务，并指定顶层模块。

随后，您可以双击任务启动仿真。下方 Run 面板会显示 Vivado 的输出

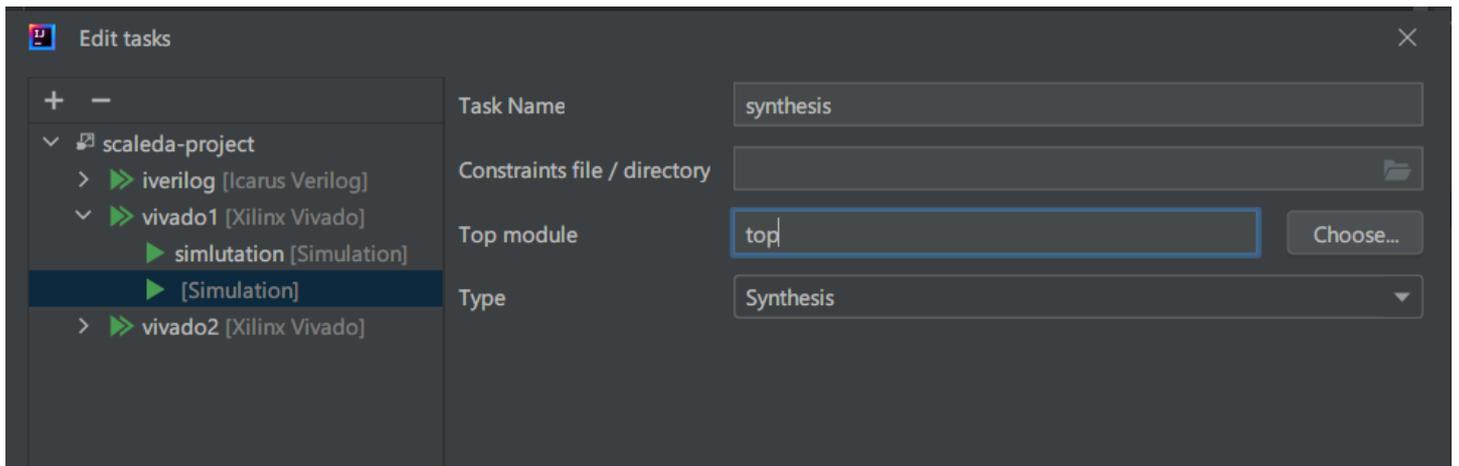


```
Time resolution is 1 ps
source testbench.tcl
## current_wave_config
$finish called at time : 40 ns : File "C:/Users/hans/Playground/scaleda-project/test/testbench.v" Line 17
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testbench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:25 . Memory (MB): peak = 460.492 ; gain = 7.852
# open_vcd "C:/Users/hans/Playground/scaleda-project/.sim/vivado1-simulation/testbench_waveform.vcd"
WARNING: [Simtcl 6-168] No object found for the given pattern.
# log_vcd [ get_object -r * ]
INFO: [Common 17-206] Exiting Vivado at Mon Apr 17 10:36:09 2023...
```

当仿真正常结束时，和 iverilog 一样，会自动弹出波形查看器。波形调试的细节请参见上文。

## 使用 Vivado 综合、实现及上板（编程）

您可以使用 Vivado 完成综合、实现等步骤。在 Vivado 目标平台下新建对应的任务，需要指定顶层模块；对于实现、上板任务，需要指定约束文件或所在的目录。



添加后，您可以在右面板中执行对应的任务。

当任务成功结束后，您可以在当前项目的 `.synth`（用于综合）、`.impl`（用于实现和编程）目录下，找到以刚刚执行的任务命名的目录。目录之下是为此任务生成的 Vivado 项目。您可以使用 Vivado 打开它。

说明：目前，综合、实现及编程的结果难以在 Vivado 之外的软件内展示。Scaleda 的最终目标是以开放格式（EDIF）作为相关硬件步骤的中间交换格式。

## IP 的使用

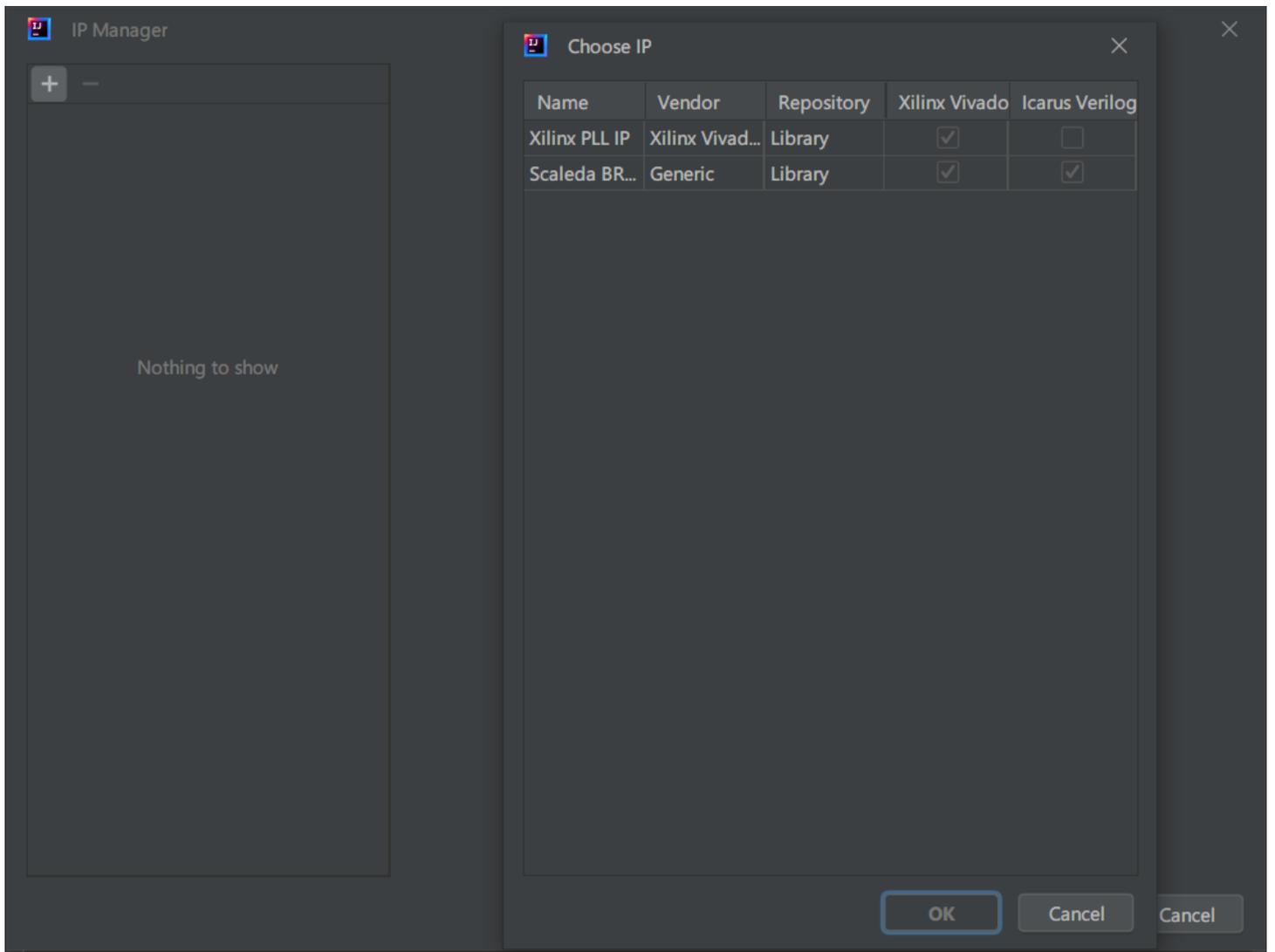
本节介绍 Scaleda 项目中 IP 核的实现。Scaleda 支持的 IP 可以分为两种：

- 通用 IP。这种 IP 在不同厂商芯片中都有功能相仿的实现，它们可以在各种平台上使用。
- 厂商 IP。这种 IP 的实现因厂商而异，因此可能无法由通用的仿真器仿真，也不能在多个平台上使用。具体的支持情况各有不同。

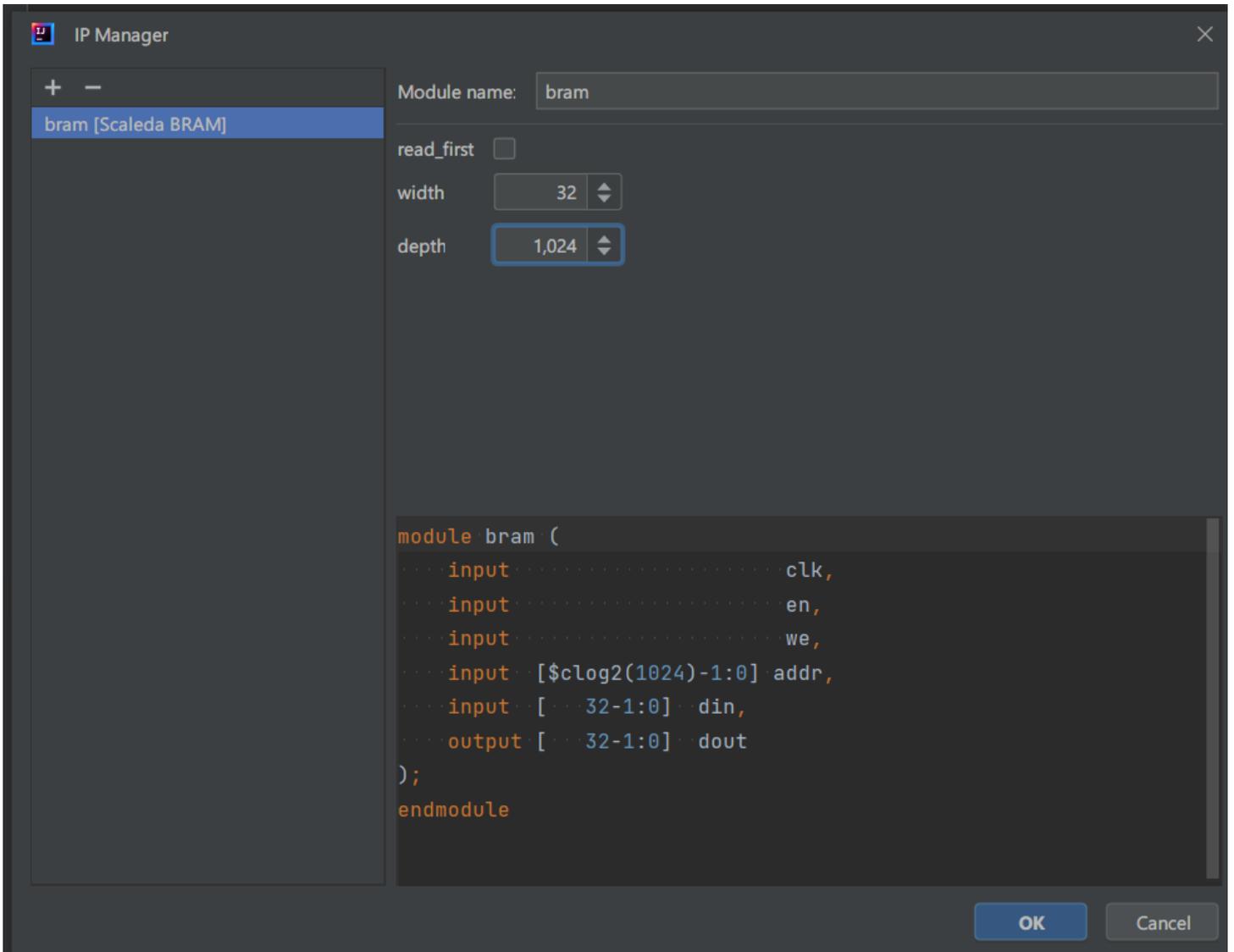
Scaleda 目前支持的 IP 有：

IP 名称	Icarus Verilog	Xilinx Vivado
通用 IP，它们在 Scaleda 中配置、加载。实际使用时，它们会对应到各厂商芯片中的硬核。		
Block RAM 同步读写的存储器	可仿真	使用 Block RAM Generator 可仿真、综合
Xilinx IP，加载时需要在 Vivado 中配置。		
PLL PLL 时钟管理器	可仿真	对应 Clock Wizard 中的 PLL 可仿真、综合

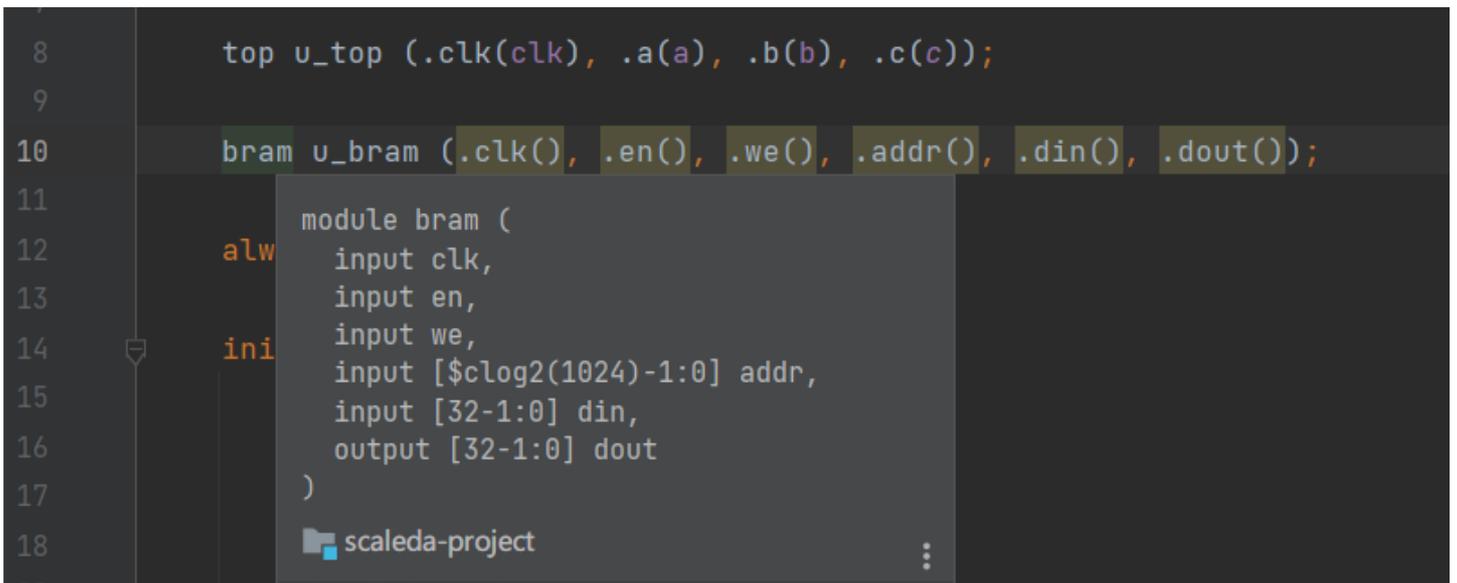
在打开 Scaleda 项目后，在右面板上方点击「IP Manager」按钮，打开 IP 编辑器，即可查看当前项目中所有的 IP。您可以点击左侧列表上方的加号，添加一个 IP。添加 IP 窗口会展示出可用的 IP 列表，以及它们能够支持的平台。



添加 IP 后，您可以设定模块名字，并设定模块参数。下方会实时预览出模块的接口情况。



随后，您可以在代码中实例化 IP 来使用。



## 愿景与反馈

本文介绍的 Scaleda 项目系统的愿景是：打造一个与厂商 EDA 软件独立的工作流，厂商 EDA 软件作为被调用的工具链，而不是工程的主体；同时，引入开源工具链如 Yosys 和 iverilog，以在一定程度上摆

脱厂商的限制，提升灵活程度。

现阶段，Scaleda 的项目系统还很不完善，且多数功能仍然是以 Vivado 为核心进行。后续，Scaleda 将引入以开放格式（EDIF）为线索的工作流程，并支持更多的工具链类型，以实现目标功能。

若您在体验中，遇到了与 Scaleda 项目相关的 bug，请向我们反馈。

## 总结与下一步

本项目的初步目标，是通过提供良好的用户体验，帮助初学者更好地了解、入门 FPGA 及至数字逻辑的世界。为实现这一目标，我们在 IntelliJ IDEA 平台上实现了一些智能化的 Verilog 编辑功能，简单实现了「Vivado 模式」以降低用户的上手成本。对于已熟悉 Vivado 使用的用户，可以把 Scaleda 作为一个增强型的编辑器使用；而对于零基础的初学者，实时的代码提示器更能帮助他们养成良好的编码习惯。不过，我们暂没有实现与 Vivado 软件本身更加紧密的对接，许多功能需要用户手动切换。同时，我们的编辑器实现也并不完美，除了存在一些体验 bug 外，还需要增加更多的代码问题检测功能。

本项目的更深层次的目标，是建立一套在厂商 EDA 之上的开发流，用户不再直接与厂商 EDA 交互，而由 Scaleda 调用工具链完成功能。我们设计了自己的项目系统，接入了两款工具链，并提供了非常简单的 IP 实现。这套系统仍然有许多完善空间，包括采用开放标准的中间格式、直观的中间提示、支持更多的工具链、更加完善的错误处理等。同时，一些功能已有设计但存在很多问题而未在本文档中展示，如远程工具链的使用等。这些问题需要更多的开发去解决，部分原有设计也需要进行优化。

限于时间仓促，Scaleda 必然存在着许多不足。非常希望您能在使用中向我们反馈所遇到的问题，包括软件本身的 bug、功能上的缺陷，以及您对开发方向的建议。