

培训 01--开发环境搭建

注意

要求在 Win 下配置，Linux 下的配置暂不涉及

Vivado 的安装

参考[这里](#)进行安装，需要注意：

- 版本暂时使用 2018.3，安装包找研一同学拷
- 选择 Design Edition 就够用了
- Tool 和 Device 建议都勾选上
- WebTalk、WinCap、System Generator 不需要
- 路径不要出现空格和中文
- 许可证可以在[这里](#)找到

检查

- 从图标启动 Vivado、Vivado HLS、SDK，检查是否能启动
- 使用 License manager 检查 License 状态

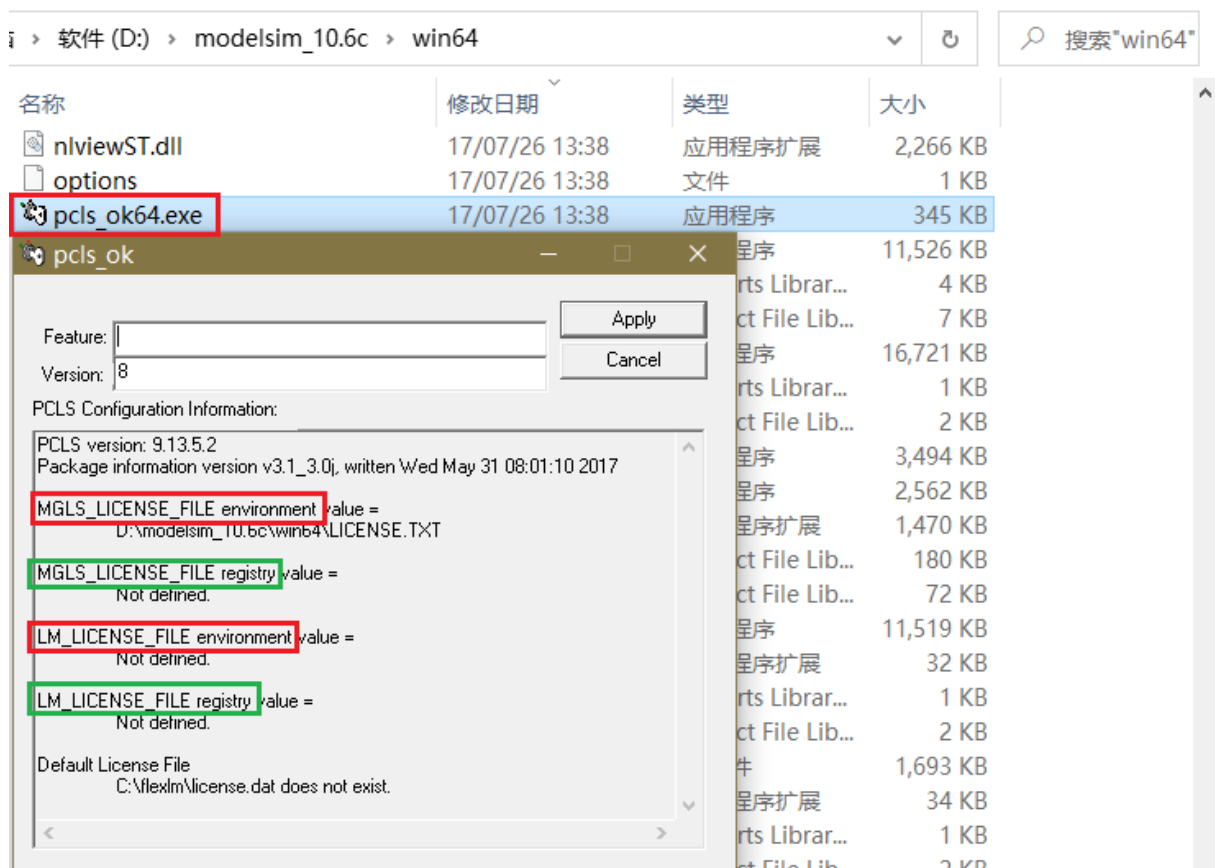
QuestaSim 的安装

参考[这里](#)进行安装，需要注意：

- QuestaSim 比 ModelSim 速度更快，更推荐
- 版本使用 10.6c(64 位)，与 Vivado2018.3 更搭配，安装包找研一同学拷
- 路径不要出现空格和中文
- 只需添加一个用户环境变量 `MGLS_LISENCE_FILE` 即可

检查

- 使用 win64 下的 pcls_ok.exe 检查配置，只要任意一项是有效的就可以
- 检查通过但仍然启动不了的建议注销或者重启机器后再尝试启动
- 同时支持从图标启动和命令行启动（在 cmd/powershell 中输入 vsim）



VsCode 的配置

对编辑器的需求

RTL 开发对编辑器的高频需求为：

- 支持语法高亮
- 支持标签跳转（变量定义，子模块定义）
- 支持模块自动例化
- 支持代码格式化
- 支持实时语法检查
- 支持列操作模式

Vim 高手可以通过配置实现上述需求，但难度较大，VsCode 下的配置相对容易些

安装版本和插件不限，满足使用要求即可，这里只是提供一种方案

本方案在 VsCode [1.54.3](#) x64 下测试正常

安装时注意把 添加到 PATH 选项勾选上，方便在命令行调用

Verilog 插件

插件名：FPGA Develop Support 0.1.10

该插件目前只能从历史版本安装，新版可能存在问题，可以在[这里](#)找到我重新封装后的包
(*kopera.fpga-support-0.1.10.vsix*)



FPGA Develop Support sterben.fpga-support
sterben | v0.1.10
Language support for TCL&HDL
[禁用](#) [卸载](#) [设置](#) 此扩展已全局启用。

[细节](#) [功能贡献](#) [更改日志](#)

FPGA Develop Support

FPGA development plugin for VS Code

If you have any questions, please post them under [issues](#). Development is not easy, please [star](#) if you like it.

[中文教程](#)

- Contains support for languages required during FPGA development (e.g.:VHDL, Verilog, SystemVerilog, TCL, XDC, SDC).
- Supports automatic testbench generation (and **save as file**).
- Supports rapid development with Vivado, **one-key new project**, **one-key synthesis**, and **one-key download**.
- The Xilinx IP of **CortexM3** is included to facilitate the rapid development of SoCs.

选择 Vivado 的编译器 **xvlog**（需要配置环境变量，后面会讲）进行实时语法检查

HDL: Document Symbols Precision

The level of detail the parser should look for when looking for symbols

full

HDL: Exclude Indexing

Exclude files from indexing by a globPattern

insert globPattern here

HDL: Force Fast Indexing

☒ Force indexing to use fast regex parsing

HDL > Linting > Iverilog: Arguments

Add Icarus Verilog arguments here (like macros). They will be added to Icarus Verilog while linting (The command "-t null" will be added by the linter by default)

HDL > Linting > Iverilog: Run At File Location

☐ If enabled, Icarus Verilog will be run at the file location for linting. Else it will be run at workspace folder. Disabled by default.

HDL > Linting: Linter

Select the verilog linter. Possible values are 'iverilog', 'verilator', 'xvlog' or 'none'

xvlog

该插件提供的能力有：

- 语法高亮（.v .sv .vhdl .tcl .xdc）
- 符号解析（定义悬停显示，子模块跳转）
- 模块自动例化（基于 python 的脚本）
- 常用 snippet 片段（代码块，文件头）
- 调用外部编译器进行实时语法检查

代码格式化

插件名：**verilog-formatter**



verilog-formatter isaact.verilog-formatter

IsaacT

v1.0.0

A Verilog code formatter using the iStyle Verilog formatter

禁用

卸载



此扩展已全局启用。

细节 功能贡献 更改日志

verilog-formatter

This is an extension for VSCode which provides a wrapper to the iStyle Verilog code formatter.

该插件可直接从插件市场安装，安装后需要配置[可执行文件](#)的路径，格式采用较紧凑的 K&R

扩展 (3)
Verilog formatter (3)

Verilog-formatter > Istyle: Args
Additional arguments to pass to istyle-verilog-formatter


Verilog-formatter > Istyle: Path
Path to the istyle-verilog-formatter binary

Verilog-formatter > Istyle: Style
iStyle predefined formatting style

K&R

波形绘制

插件名：Waveform Render

 **Waveform Render** | bpenuelas.waveform-render
Borja Penuelas | v0.18.0
Draw timing diagram waveforms with WaveDrom inside VSCode
[禁用](#) [卸载](#) [此扩展已全局禁用](#)

细节 功能贡献

Render waveforms with WaveDrom inside VSCode

:page_with_curl: Open a JSON file containing a WaveDrom waveform, like

```
{ signal: [
  { name: "clk",      wave: "p.....|..." },
  { name: "Data",     wave: "x.345x|=..x", data: ["head", "body", "tail", "data"] },
  { name: "Request",  wave: "0.1..0|1.0" },
  {} ,
  { name: "Acknowledge", wave: "1.....|01." }
]}
```

该插件可直接从插件市场安装，创建一个.json 文件即可开始绘制波形，参考[教程](#)

建议

- 每个信号的 wave 域写在最前面，方便对齐（结合 VsCode 的列模式很方便）
- json 文件不支持注释，但可以自己定义域段插入注释，增加可维护性

待办管理

插件名：Todo+

RTL 开发不同于软件开发，不能随时调试，从编码到验证的周期较长，即使前期的设计方案比较明确，实际编码时也会有所调整。

编码时建议明确任务，哪些是已经实现的，哪些还没有实现，哪些实现可能有问题，哪些地方出现了 bug 需要 fix。

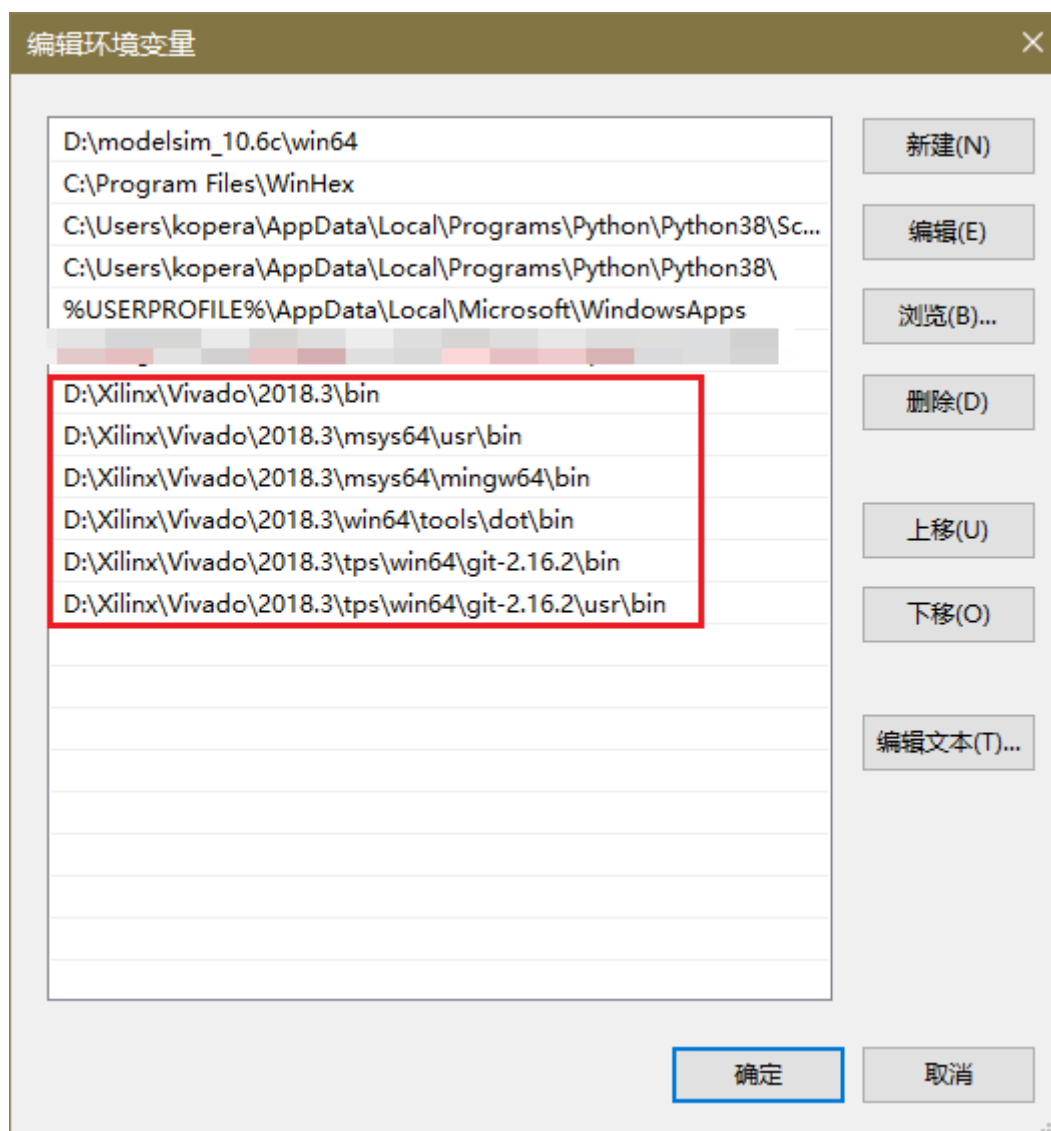
一方面在代码中插入各种标签记录该模块的进度，另一方面每个工程下维护一 TODO 文件记录总体的任务情况。

这个插件有兴趣的同学自行学习。

进阶配置

GNU 环境配置

安装了 Vivado 后，为 Path 环境变量添加以下 6 个条目，（路径根据实际替换）

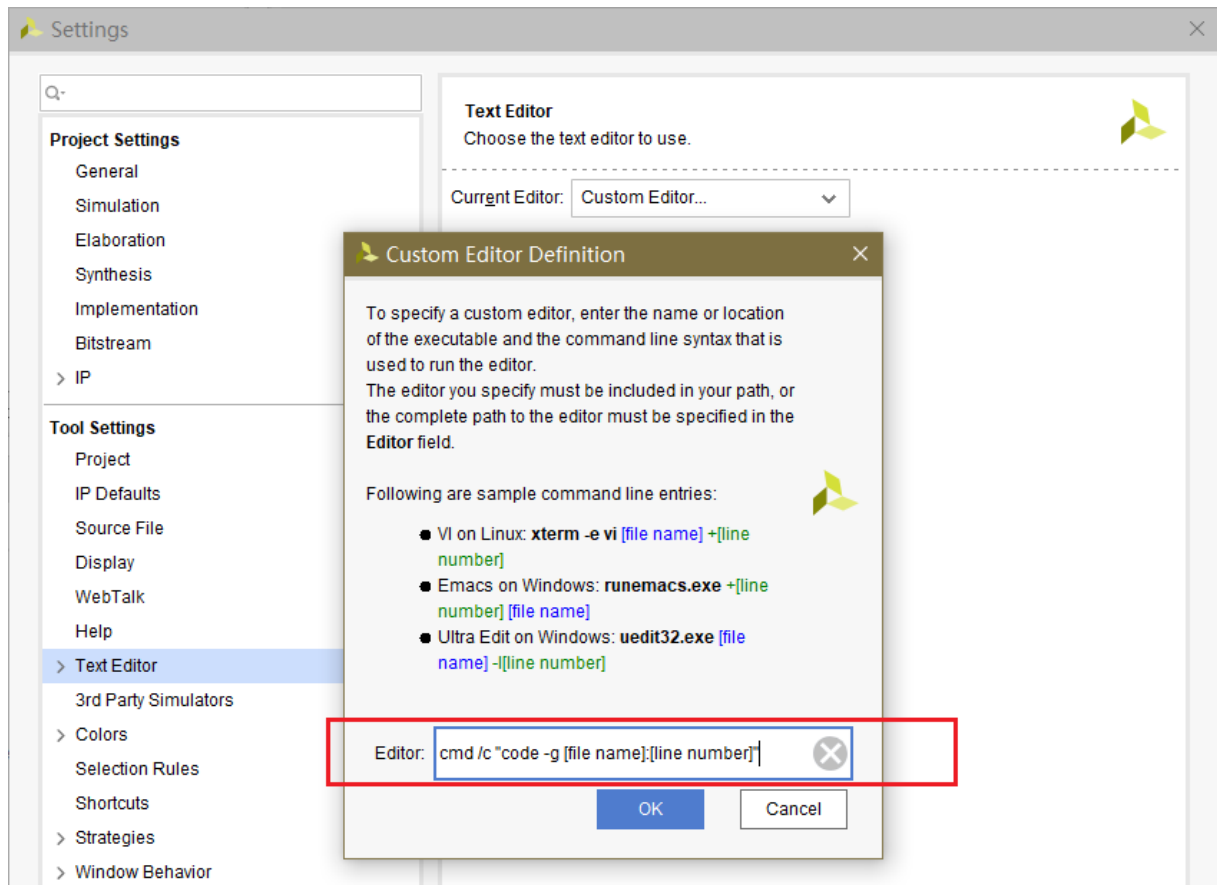


就可以在 cmd 中访问 xvlog、xelab、xsim 以及 make、mkdir、gcc、dot、git、perl 等程序，便于编写脚本。

使用外部编辑器

使用 Vivado 自带仿真器时并不建议这样做，因为第三方编辑器不支持仿真时的断点设置、变量值显示等功能。

使用外部仿真器时则建议也使用更强的的外部编辑器，以 VS Code 为例，



```
cmd /c "code -g [file name]:[line number]"
```

检查

为了便于后续的开发，要求大家完成进阶配置，并进行下列检查

- 命令行调用 xvlog、xelab、xsim
- 命令行调用 vlog、vlib、vmap、vsim
- 命令行调用 code、code -g
- 命令行调用 make、mkdir、touch、sed、awk、gcc
- 命令行调用 perl、git、dot