GitHub 团队协作操作指南

欢迎加入我们的 FPGA 比赛项目！这篇指南帮助你快速上手 GitHub，完成代码开发、提交和团队协作。仓库已创建好，你只需要按照以下步骤操作。

1. 准备工作

1. 安装 Git

   ：

   - 下载 Git（https://git-scm.com/downloads），安装后命令行输入 `git --version` 确认。
   - 可选：用 GitHub Desktop（https://desktop.github.com/）或 VS Code 的 Git 插件，简化操作。
2. GitHub 账户

   ：

   - 确保有 GitHub 账户，分享你的用户名以加入仓库协作者。
3. 配置 Git

   （首次使用）：

   ```
   git config --global user.name "你的名字"
   git config --global user.email "你的 GitHub 邮箱"
   ```

2. 获取仓库代码

1. 克隆仓库

   ：

   - 获取仓库 URL（形如 `https://github.com/用户名/仓库名.git`）。
   - 命令行：`git clone 仓库URL`（如 `git clone https://github.com/用户名/FPGA-Competition.git`）。
   - GitHub Desktop：添加仓库 > Clone repository > 粘贴 URL。
   - 这会下载仓库到本地，生成一个项目文件夹。

3. 基本开发流程

每次开发新功能或修复问题，按以下步骤：

1. 更新本地仓库

   ：

   - 在项目文件夹运行：`git pull origin main`（确保本地代码与 GitHub 主分支同步）。
2. 创建分支

   ：

   - 不要直接改 main 分支！为你的任务创建新分支：

     ```
     git checkout -b 分支名
     ```

   - 分支命名建议：`feature/功能名`（如 `feature/fpga-uart`）或 `bugfix/问题名`。

3. 修改代码

   :

   - 在本地编辑 FPGA 代码（Verilog/VHDL 等）。
   - 注意：不要提交编译输出文件（如 .log、build/），这些应在 `.gitignore` 中忽略。
4. 提交变化

   :

   - 添加修改：`git add .`（所有文件）或 `git add 文件名`（单个文件）。
   - 提交：`git commit -m "简述你的修改，如 '实现 UART 模块'"`。
5. 推送到 GitHub

   :

   - 推送分支：`git push origin 分支名`（如 `git push origin feature/fpga-uart`）。

4. 提交代码给团队（拉取请求）

1. 创建拉取请求 (Pull Request, PR)

   :

   - 打开 GitHub 仓库页面，点击 "Pull requests" > "New pull request"。
   - 选择 `base: main`（目标分支）和 `compare: 你的分支`。
   - 输入标题（如 "添加 UART 模块"）和描述（说明改了什么）。
   - 点击 "Create pull request"。
2. 等待审查

   :

   - 团队成员会审查你的代码，可能会提出修改建议。
   - 如果需要调整，继续在同一分支 commit 和 push，PR 会自动更新。
3. 合并

   :

   - 审查通过后，PR 会被合并到 main 分支。

5. 处理冲突

- 如果

  ```
  git pull
  ```

  提示冲突（多人改了同一文件）：

  1. 打开冲突文件，查找 `<<<<<<<` 和 `>>>>>>>` 标记。
  2. 手动保留正确代码，删除标记。
  3. 运行 `git add 文件名` 和 `git commit`。
- 建议：开发前先 `git pull origin main`，减少冲突。

6. 团队协作工具

- Issue

  ：看到任务或问题，查看仓库的 "Issues" 页面。可以用评论讨论。

  - 认领任务：留言或请求指派。
- **通知**：GitHub 会通过邮件或通知提醒 PR 和 Issue 更新，及时查看。

- **同步代码**：每次开发前运行 `git pull origin main`，确保代码最新。

7. 最佳实践

- **小步提交**：每次 commit 只包含一个功能或修复，描述清晰。
- **分支命名**：用 `feature/` 或 `bugfix/` 前缀，方便管理。
- **不要提交无关文件**：如 Vivado/Quartus 的临时文件（已在 `.gitignore` 忽略）。
- **沟通**：有问题在 Issue 或团队群聊中讨论，避免直接改 main 分支。

8. 常见命令速查

```
git clone 仓库URL          # 克隆仓库
git pull origin main       # 更新本地代码
git checkout -b 分支名     # 创建并切换分支
git add .                  # 添加所有修改
git commit -m "描述"       # 提交变化
git push origin 分支名     # 推送分支
git status                 # 查看当前状态
```

9. 遇到问题

- 报错或不明白的地方，截图错误或问题，联系团队或查阅：
  - GitHub 帮助：https://docs.github.com/en
  - 简单教程：搜索 "Git 教程 廖雪峰" 或 Bilibili 的 Git 视频。
- FPGA 项目注意：确保代码模块化，方便多人协作。

开始试试克隆仓库和提交一个简单 PR 吧！有问题随时问团队。