技术工作流 • 参考指南

FPGA 与嵌入式开发辅助工作流

FPGA 与嵌入式独立闭环 · 面向研发过程的开发辅助

Xg-Stellar 面向研发过程提供两套独立能力。FPGA 与嵌入式只共享桌面应用、模型配置、知识库、文件浏览和日志基础设施；它们不共享业务规格、任务状态、执行控制或完成结论。

FPGA 开发辅助

1. 工程识别

打开 FPGA 工程后，只读识别 RTL、测试台、约束、仿真脚本、波形和顶层模块候选。此阶段不修改文件，并明确列出缺失的工程信息。

2. 开发任务整理

将用户需求整理为本轮 FPGA 开发记录，包括目标模块、接口、时钟、复位、位宽、关键时序、测试场景、预计修改文件和待确认问题。关键接口语义不明确时先请用户确认；简单任务允许使用明确标注的开发假设。

3. 读取工程基线

修改已有设计前读取当前 RTL、模块引用、测试台和最近验证记录。优先修改真实文件，不另建重复实现；保留修改差异，不改动任务外文件。

4. 生成或修改

新任务生成 RTL 和必要测试台；修改任务在现有文件上完成；缺陷任务先复现再修复。工作文件写入 `FPGA/rtl/` 和 `FPGA/testbench/`，写入只表示待验证，不等于通过。

5. 仿真与检查

编译并运行测试台，收集通过、失败、超时和工具错误，按需生成波形。根据任务执行 RTL 静态检查和跨时钟域检查。报告必须分别列出已执行、未执行和失败的项目。

6. 修复循环

根据报告判断问题属于 RTL、测试台还是环境。RTL 错误修改 RTL；只有测试台与已确认行为冲突时才修改测试台；环境缺失时停止自动修复。每轮重新执行已启用项目，达到迭代上限后保留诊断并交还用户。

7. 开发结果

结果使用“开发验证通过”“基础仿真通过”“需要继续修复”“环境未就绪”或“等待用户确认”。输出修改文件、仿真结果、检查项、波形路径、未执行项和剩余风险。本轮记录保存在 `FPGA/验证/<runId>/`。

嵌入式开发辅助

1. 工程识别

打开嵌入式工程后，只读识别 C/C++ 源码、头文件、测试、构建配置、目标平台和可用测试环境，不读取 FPGA 任务或状态。

2. 开发任务卡

将需求整理为独立嵌入式开发任务卡，包括模块、公共 API、输入输出、边界条件、错误处理、时间行为、目标平台、测试场景和待确认项。接口、时间单位或目标环境不明确时先确认。

3. 读取模块基线

定位头文件、实现、依赖、已有测试和构建配置。磁盘现有文件优先；公共 API 变更需提示影响；只修改当前任务涉及的模块。

4. 生成或修改

新模块生成头文件、实现和基础测试；现有模块保留兼容接口并更新实现与测试；空骨架只有在需求明确时才补全。硬件依赖通过模拟层隔离，主机单测不冒充真实板卡结果。

5. 测试、覆盖率和静态分析

编译运行单元测试并区分编译、链接、断言、崩溃、超时和环境问题。默认统计覆盖率并报告未覆盖代码；默认执行静态分析并报告内存、越界、未初始化、整数转换和资源管理风险。

6. 修复循环

实现错误修改实现后重跑相关测试、覆盖率和静态分析。只有测试与明确需求或公共接口冲突时才能修改测试。环境问题停止代码修复；需求问题回到任务卡澄清。

7. 可选板卡调试

对话流程只能检查清单、产物哈希和调试条件。真实烧录必须由用户在板级闭环页面确认，并保留客户机许可。烧录、串口和启动标识结果只作为开发调试记录。

8. 开发结果

输出修改源码、头文件、测试文件、测试结果、覆盖率、静态分析、板卡调试状态、未执行项和剩余风险。嵌入式记录只写入嵌入式工程。

隔离规则

1. FPGA 不创建、读取或更新嵌入式任务卡和完成状态。
2. 嵌入式不创建、读取或更新 FPGA 任务记录和验证结果。
3. 一个领域失败不会改变另一个领域的状态。
4. 知识库资料可分别引用，但不会生成跨领域统一规格。
5. 面向用户只表达开发辅助结果，不表达工业交付、合规认证或正式验收结论。