

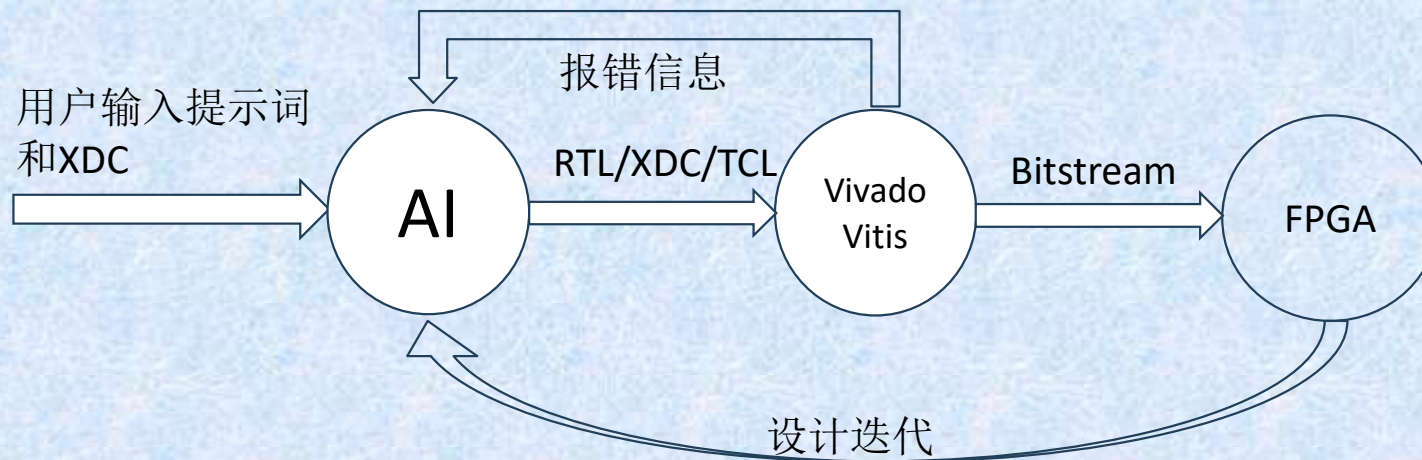
AI 助力 Xilinx FPGA 开发

上海科技大学 张炀杰

目录： AI 辅助 AMD Xilinx FPGA 开发

- AI 辅助 AMD Xilinx FPGA 开发的流程
- 推荐的工具、模型
 - 免费在线对话模型
 - 本地模型和 Agent 工具
 - 付费闭源模型/Agent 工具
- 用 LLM 生成简单的 RTL 项目：以 [Boolean Board Digital Logic Course](#) 为例
 - 提示词工程
 - 生成 ALU

AI 辅助 AMD Xilinx FPGA 开发流程



- AMD Xilinx 工具链提供了 Vivado 和 Vitis 用于 RTL 和 HLS 的开发
 - Vivado 和 Vitis 可以使用 TCL 脚本进行控制
 - 可以通过 TCL 脚本控制 Vivado/Vitis
 - 可以让 AI 根据报错信息进行调试
 - 人工观察上版行为后，将现象反馈给 AI，进行设计迭代
 - 可以搭建 agent 来自动化实现 AI 与 Vivado/Vitis 间的循环

AI 辅助 AMD Xilinx FPGA 开发：推荐工具

- 免费（在线使用或本地部署，效果不错但达不到最好）：
 - Deepseek 系列：推荐使用 v3.1
 - Qwen3, Qwen3-Coder 等模型：最大尺寸的 Qwen3 Coder 可以与 Claude 4 媲美，但硬件要求高
 - 优先使用 Coder 模型或最新一代模型，例如 [Qwen3-Coder: 在世界中自主编程 | Qwen](#)
 - Cline, Trae 等 AI IDE + 本地模型
 - Qwen-Code, Gemini-CLI 等 Agent 编码工具 + 本地模型

AI 辅助 AMD Xilinx FPGA 开发：推荐工具

- 付费闭源模型/Agent 工具（效果好但需要付费/科学上网）
 - Claude4 + Claude Code 编程领域最佳模型，但国内手机号无法注册
 - Gemini Pro 2.5 + Gemini CLI 效果略逊于 Claude/Claude Code，但有超长的上下文窗口
 - Qwen-Code + Qwen Coder：基于 Gemini CLI 二次开发，有免费额度限制，每日 2000 次免费调用
 - Cursor IDE：付费的 AI IDE，可以使用多种在线模型，内置 Agent 模式
 - 相比于单一的模型，Agent 能够进行工具调用，并处理超出传统模型上下文长度的大型代码库

AI 辅助 AMD Xilinx FPGA 开发：推荐工具

- 如果硬件条件允许且有本地部署需求，可在本地部署最新版本的，参数量尽可能大的模型
 - 代码任务对精度要求较高，如在本地部署，建议选择 Q4_K_M 或更高精度量化的版本，同时参数量尽可能大
 - 可以通过本地中小模型 + 类似 Claude Code 的 Agent 工具 + 知识库来在性能和安全/隐私性之间做平衡
 - 例：128G 统一显存的 AMD AI MAX 395+ 可以本地以 40+token/s 的速度运行 Qwen3 30B A3B 的 Q8_0 量化版本

用 LLM 生成简单的 RTL 项目

- 从 [Digital Logic | Real Digital](https://www.realdigital.org/doc/c04fb589c6830cfe0dd7b9aef5053987) 这个课程中选取作业作为例子，说明如何使用 LLM 辅助学习 Xilinx FPGA 开发
- 可以尝试让 LLM 总结课程内容
 - 例：使用网页端 Qwen 3-235B-A22B-Instruct-2507 对 Introduction and Overview 进行总结

根据你提供的网页内容 (<https://www.realdigital.org/doc/c04fb589c6830cfe0dd7b9aef5053987>)，这一章节本身并没有具体的、需要你完成的“设计任务”或“项目作业”。

这个网页是《数字逻辑课程》的“介绍与概述” (Introduction and Overview)，它的作用是：

- 告诉你这门课是什么。
- 解释你需要什么工具和材料。
- 指导你应该如何学习和完成后续的项目。
- 设定学习的心态和目标。

★ 所以，你在这个“章节”里需要完成的任务是：

这些任务是为开始并成功进行后续的实际项目所做的准备和规划：

- 理解课程结构和目标：
 - 明白这门课是通过一系列动手实践项目来学习的。
 - 理解课程强调自主学习、实践和解决问题。
- 准备必要的硬件和软件 (这是最关键的行动项)：
 - 获取一块 Boolean Board：这是课程专用的 FPGA 开发板。你需要购买或准备好它。
 - 下载并安装 Xilinx Vivado Webpack：这是免费的、功能完整的数字电路设计软件。你需要从 Xilinx 官网下载并安装它。
- 了解评估方式：
 - 知道你的项目完成后需要在开发板上演示。
 - 了解评分标准 (0-4分)，明白“完成所有要求”是拿3分 (通常视为基础满分)，“代码整洁、设计优秀”可以拿4分。
- 掌握正确的学习方法：
 - 阅读后续项目的“背景知识” (绿色按钮链接的文档)。
 - 认真完成“设计要求” (Design Requirements)，这是必须完成的核心任务。
 - 强烈建议尝试“设计挑战” (Design Challenges)，虽然可选，但对提升能力非常有帮助。
 - 学会使用仿真器 (Simulator)：这是独立验证你设计是否正确核心工具。
- 规划下一步行动：
 - 这个章节的结尾明确告诉你：“Refer to the first project...” (参考第一个项目...)。
 - 你的下一个真正任务是：进入课程的第一个项目 (Project 1)，按照里面的步骤开始创建你的第一个 Vivado 工程，并将设计下载到你的 Boolean Board 上运行。

总结

简单来说，这个“章节”本身没有需要你提交的作业。它是一份学习指南和准备清单。

✅ 你现在最需要完成的任务是：

- 准备好 Boolean Board。
- 下载并安装 Xilinx Vivado Webpack。
- 然后，立刻去学习并完成 课程的第一个实际项目 (Project 1)。

一旦你完成了这些准备工作，你就真正踏上了数字逻辑设计的学习之旅。



用 LLM 生成简单的 RTL 项目

- 如果我们要让 LLM 针对某一款板卡做 RTL 设计，我们需要让它知道板卡上有哪些外设引脚，对应的编号是什么
 - 对于 **Boolean Board**，我们可以通过将官方模板 XDC 文件（包含了全部引脚信息的约束文件）发送给 LLM 的方式使其知晓板卡的硬件信息
 - 如果没有类似文件，需要提前告知模型，需要绑定哪些引脚

用 LLM 生成简单的 RTL 项目

- 如何写一个好的提示词？
 - 可以让 AI 帮改
 - 以点亮 Boolean Board 的 LED 为例

🔗 **代码模式** 请你帮我修改提示词：“我需要修改针对 Boolean Board 的以下 Verilog 设计以点亮所有的 LED，你可以从我上传的 xdc 文件了解 Boolean Board 的可用硬件资源”



好的，我们可以将您的提示词修改得更清晰、具体一些。修改后的提示词可以是：

“请帮我修改提供的 Verilog 代码，目标是让 Boolean Board 上的所有 LED 灯都点亮。请参考我上传的 .xdc 约束文件来了解 Boolean Board 的具体硬件资源（例如 LED 和引脚的对应关系）。”

修改说明：

- **明确目标：**“点亮所有的 LED” 改为 “让 Boolean Board 上的所有 LED 灯都点亮”，更具体。
- **明确依据：**“从我上传的 xdc 文件了解 Boolean Board 的可用硬件资源” 改为 “参考我上传的 .xdc 约束文件来了解 Boolean Board 的具体硬件资源（例如 LED 和引脚的对应关系）”，更清晰地说明了 .xdc 文件的作用和需要关注的信息。

这样的提示词能更有效地传达您的需求。不过，要真正帮您修改 Verilog 代码，我还需要您提供：

- 您当前的 Verilog 设计代码。
- 您提到的 Boolean Board 的 .xdc 约束文件内容（至少包含 LED 相关的约束）。

如果您能提供这两份文件的内容，我就能根据您的要求和硬件信息，为您生成修改后的完整 Verilog 代码 artifact。



用 LLM 生成简单的 RTL 项目

- 即使是点亮 LED 灯这样的简单任务，本地小模型（参数量小于 10b 的模型）如果不经特殊调整，也很难完成编程任务
- 即使给了 XDC 文件，也数不明白板子上有几个 LED，几个开关
- 而即使是当下最前沿的闭源模型，也无法光凭 XDC 文件就知道硬件资源的物理排列方式
 - 例：不知道七段发光数码管哪个是高位，哪个是低位

用 LLM 生成简单的 RTL 项目

- 用 LLM 直接生成个 ALU: [Project 10 A Simple ALU](#)
 - Claude-3.7-Sonnet 能够成功生成 RTL 代码和 XDC 约束文件, 但课程要求用到 8 个七段发光数码管中的四个, LLM 生成的代码只用到了 0, 1 和 4, 5, 并且 0 号发光数码管始终为 0
 - 高低位弄反了 + 无法认识物理世界

Create an 8-bit ALU in Verilog that supports the functions shown in the table. Connect data registers to both ALU inputs, and load the data registers from slide switches, using push buttons for load enable signals, as shown. Connect three slide switches to the ALU control input, and connect the ALU output to your seven-segment display controller through the Bin2BCD converter. Program data into the registers, and demonstrate all ALU functions. You can use `boolean.xdc` as reference for the pin information on this board.

opcode	description	output F
000	Addition	$A + B$
001	Increment	$A + 1$
010	Subtract	$A - B$
011	Bit-wise XOR	$A \oplus B$
100	Bit-wise OR	$A B$
101	Bit-wise AND	$A \& B$

用 LLM 生成简单的 RTL 项目

- 用 LLM 直接生成个 ALU: [Project 10 A Simple ALU](#)
 - Claude-3.7-Sonnet 能够成功生成 RTL 代码和 XDC 约束文件, 但课程要求用到 8 个七段发光数码管中的四个, LLM 生成的代码只用到了 0, 1 和 4, 5, 并且 0 号发光数码管始终为 0
 - 高低位弄反了 + 无法认识物理世界

Create an 8-bit ALU in Verilog that supports the functions shown in the table. Connect data registers to both ALU inputs, and load the data registers from slide switches, using push buttons for load enable signals, as shown. Connect three slide switches to the ALU control input, and connect the ALU output to your seven-segment display controller through the Bin2BCD converter. Program data into the registers, and demonstrate all ALU functions. You can use `boolean.xdc` as reference for the pin information on this board.

opcode	description	output F
000	Addition	$A + B$
001	Increment	$A + 1$
010	Subtract	$A - B$
011	Bit-wise XOR	$A \oplus B$
100	Bit-wise OR	$A B$
101	Bit-wise AND	$A \& B$

总结

- 通过 TCL 脚本，可以让 AI 操控 Vivado 工具，生成 RTL、XDC 和 TCL 脚本，根据错误信息进行自动化 Debug
 - 编码时优先选择高精度的大模型，结合 Agent 编码工具效果更佳
 - 需要将可用的硬件资源告诉 LLM
 - 但上板验证还需要人类介入
- LLM 的编码性能飞速提升，但 FPGA 上的 RTL 开发不仅仅是单纯的编码
 - 需要人类介入，将物理世界的情况告知 LLM，并观察上板后的现象