中国科学院大学
University of Chinese Academy of Sciences

基于大语言模型的硬件自动化设计与验证

赵地、齐洪钢
中国科学院大学

中国科学院大学
University of Chinese Academy of Sciences

大纲

一、硬件设计与验证现状

二、大语言模型的潜力

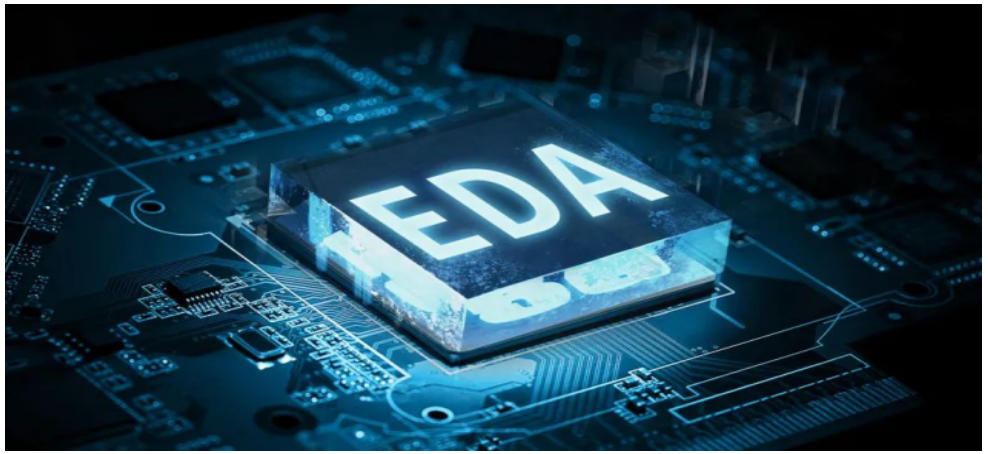
三、LLMs在代码生成中的应用

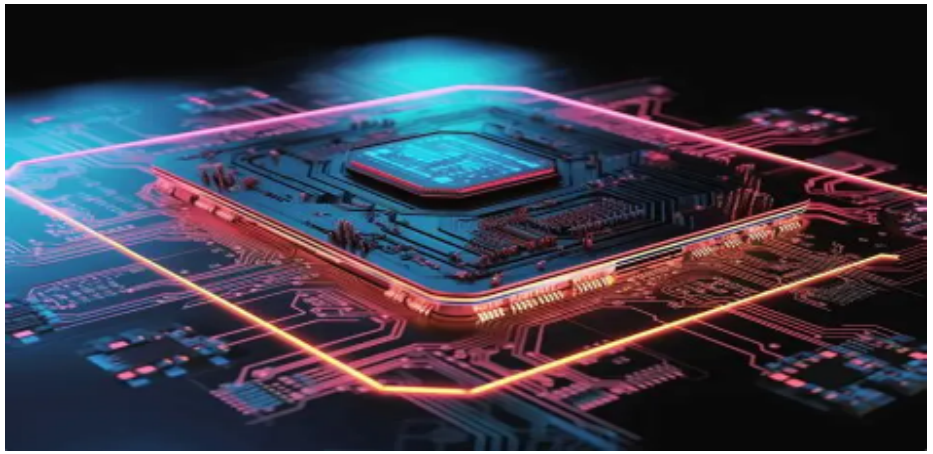
四、LLMs模型在验证中的应用

五、总结

一、硬件设计与验证现状

- 硬件设计复杂度高
  - 硬件设计是一项复杂且高成本的工程任务，随着芯片复杂度提升，硬件设计涉及多层级模块协同、严苛时序约束及多物理域交叉，传统电子设计自动化（EDA）工具耗费大量时间，设计周期延长。
- 硬件验证成本高且有错误率
  - 数十亿晶体管的芯片设计，验证阶段可能需耗费两倍于设计的时间，导致项目进度延迟和成本超支。
  - 面对超大规模集成电路时，难覆盖所有边缘案例，错误风险增加。





二、大语言模型的潜力

- 代码生成
  - 大语言模型（LLMs）的出现为自动化硬件描述语言（HDL）代码生成和提供了新思路。例如，ChatGPT能够将自然语言描述转化为Verilog代码，但现有研究未充分验证其可靠性和效率。
  - 设计者通过描述模块功能，LLMs自动生成对应的Verilog代码，减少人工编码错误，有望显著提升硬件设计的效率和质量。
- 形式化验证
  - LLMs生成形式化证明辅助工具，结合传统工具提高形式化验证的效率和成功率。
  - LLMs自动生成测试用例和验证脚本，大幅减少人工编写工作量，缩短验证周期。
  - 在芯片设计验证中，LLMs辅助生成形式化证明脚本，与现有工具协同，成功证明更多复杂设计的正确性。

三、LLMs在代码生成中的应用 — 框架与模型

ChipGPT：多阶段生成与优化

ChipGPT通过多阶段生成Verilog代码，首先生成初稿，再通过优化满足性能、功耗和面积（PPA）约束。其生成的代码较ChatGPT- 3.5更简短且面积更小，但可扩展性受限。
例如，生成的8位加法器面积减少15%，但仅适用于中小规模设计，对于大规模复杂设计的适应性有待进一步提升。

RTLLM：基准测试集与自规划技术

RTLLM是开源基准测试集，包含30个设计问题，结合自规划（self- planning）技术提升生成质量。例如，LLMs先分解任务（如设计计数器），再逐步生成子模块代码，但需人工提供测试用例。
通过统一的问题描述和测试用例，为不同LLMs提供公平评估环境，但覆盖的设计类型有限，如缺少高速接口模块。

AutoChip：仿真反馈循环与错误修正

AutoChip集成仿真反馈循环，通过Icarus Verilog检测错误并迭代修正。在HDLBits基准测试中提升24.2%正确率，但对复杂问题（如状态机）效果有限。
例如，检测到波形不匹配后，LLMs重新生成代码，但复杂问题仍需人工干预，自动化程度有待提高。

GPT4AIGChip：分层生成策略与AI加速器设计

GPT4AIGChip针对AI加速器设计，采用分层生成策略。
例如，将ResNet分解为卷积层、池化层等模块，逐个生成并集成，生成的设计在延迟上优于人工方案。
采用模块化生成策略，降低单次生成复杂度，但依赖目标工艺库的数据，对不同工艺的适配性需进一步优化。

三、LLMs在代码生成中的应用 — 关键挑战

01

代码正确性问题

LLMs生成的代码常存在语法错误或逻辑错误，导致代码无法正常工作或功能偏差。例如，生成的状态机缺少关键状态转移，导致功能失效，需要结合形式验证工具进行自动化检查和修正。

02

领域数据缺乏

公开的Verilog数据集规模远小于Python，限制模型训练效果。硬件设计领域的数据稀缺，导致模型对硬件设计规范（如时钟域交叉处理）理解不足。例如，硬件设计中特定类型的漏洞数据稀缺，影响模型泛化能力，需要构建高质量的硬件设计数据集以提升模型性能。

03

PPA优化不足

多数研究未考虑生成代码的综合结果（如面积、时序），难以评估实际硬件性能。生成的代码可能在功能上正确，但在实际硬件实现中无法满足性能、功耗和面积的要求。例如，未优化的代码可能导致芯片面积过大或时序不满足要求，需要开发专门的优化算法和工具来提升生成代码的PPA指标。

四、LLMs在验证中的应用 — 断言生成

01

自然语言到硬件断言的转换

LLMs将自然语言描述的硬件设计属性转化为SystemVerilog断言，实现自动断言生成。

例如，将“寄存器值不能超过100”转化为对应的SystemVerilog断言代码，准确率达80%以上。

02

断言生成中的上下文重要性

提供详细上下文信息可显著提高LLMs生成断言的准确性。

在复杂芯片设计中，详细描述设计功能和约束后，LLMs生成的正确断言比例从30%提升至70%。

03

断言生成的评估与改进

通过模拟器评估生成断言的有效性，结合反馈优化LLMs生成策略。

某项目中，模拟器反馈显示LLMs生成的断言有20%未触发，经优化后，触发率提升至90%。

四、LLMs在验证中的应用 — 覆盖率关闭

基于LLMs的代码覆盖率预测

LLMs根据代码和测试用例预测代码覆盖率，辅助验证团队优化测试策略。

在Python代码验证中，LLMs预测覆盖率与实际覆盖率偏差在5%以内，为测试优化提供可靠依据。

覆盖率预测的准确性提升

使用少量示例引导LLMs预测，可显著提高其准确性。

在多语言代码验证中，采用少量示例后，LLMs预测准确率从60%提升至85%。

覆盖率预测的应用拓展

将LLMs覆盖率预测应用于硬件描述语言，为硬件验证提供新思路。

在Verilog代码验证中，LLMs预测覆盖率与实际值偏差控制在10%以内，有效辅助硬件验证。

四、LLMs在验证中的应用 — 形式化验证

LLMs在形式化证明中的应用

LLMs生成形式化证明辅助工具，结合传统工具提高形式化验证的效率和成功率。
在芯片设计验证中，LLMs辅助生成形式化证明脚本，与现有工具协同，成功证明更多复杂设计的正确性。

形式化验证中的LLMs性能优化

通过优化LLMs训练数据和参数，提高其在形式化验证中的性能。
在软件形式化验证中，优化后的LLMs证明成功率提升20%，验证时间缩短30%。

形式化证明中的LLMs错误修复机制

LLMs通过分析失败证明的错误信息，自动修复并重新生成证明，提高证明成功率。
在数学定理证明中，LLMs结合错误修复机制，证明成功率从30%提升至60%。

四、LLMs在验证中的应用 — 关键挑战

- 性能提升
  - 训练数据对LLMs性能的影响：更丰富的训练数据可显著提高LLMs在验证中的性能。
  - 模型架构对LLMs性能的影响：优化模型架构可进一步提升LLMs的验证性能。在自然语言处理任务中，优化模型架构后，LLMs性能可提升30%，效率可提升20%。
- 可靠性提升
  - 提升LLMs在验证中的可靠性是确保验证结果可信的关键，其中采用多种策略如多模型验证等是提升LLMs的可靠性的关键措施。未来，通过引入可信执行环境等技术，LLMs在高安全需求验证中的可靠性有望达到95%以上。
- 应用扩展
  - LLMs在新兴验证领域如量子计算验证中展现出巨大潜力。在量子算法验证中，LLMs辅助生成验证方案，验证效率提升40%。
  - 面对新应用领域的挑战，需不断优化LLMs技术和应用策略，并且将LLMs与区块链等技术融合，可拓展其在验证中的应用范围。

五、总结 — 研究结论

- LLMs在代码生成和验证中展现潜力，但需解决正确性、安全性和可扩展性问题。现有技术多针对小规模设计，缺乏对复杂场景的支持，限制了其在硬件设计中的广泛应用。
  - 在代码生成方面
    - LLMs生成的代码在功能正确性上存在不足，功能正确性上常忽略硬件特有的时序逻辑、资源约束等细节，导致生成代码与硬件架构不匹配，需要进一步优化和改进，以满足实际硬件设计的需求。
  - 在验证方面
    - 现有LLMs验证机制对复杂硬件场景的覆盖度严重不足，停留在表层语法校验或单一功能模块的简单测试。硬件设计需要持续的迭代优化，验证工具需能根据代码修改自动调整测试用例，但LLMs生成的验证方案往往缺乏动态适配能力，在面对模块接口变更等情况时，容易出现测试覆盖盲区，无法支撑大规模硬件开发的全生命周期验证需求。

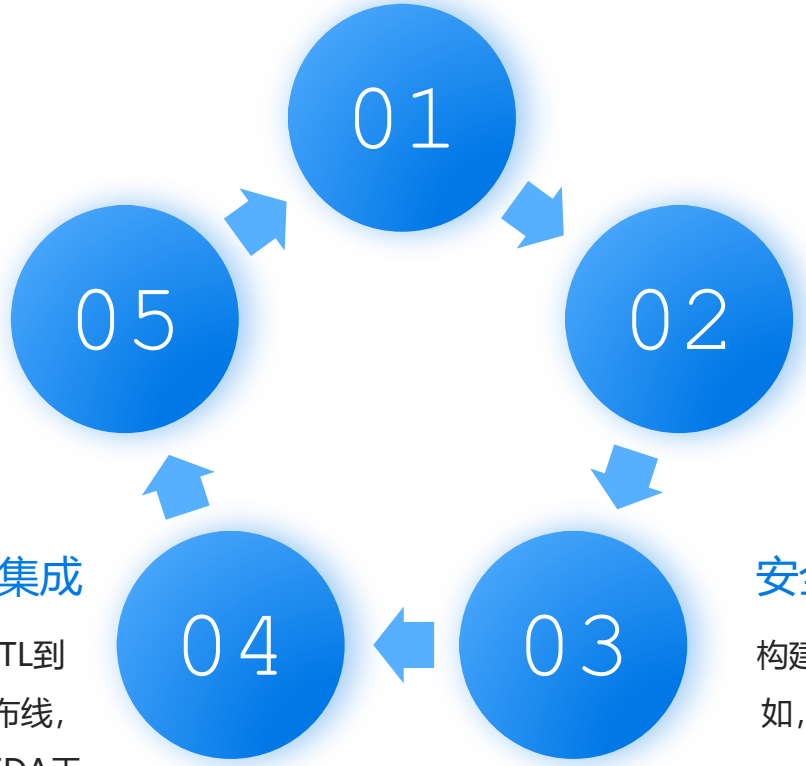
五、总结 — 未来方向

架构规范自动化

利用LLMs将自然语言需求转化为设计文档，减少人为错误。例如，将用户需求（如“低功耗传感器接口”）转化为详细设计文档，减少沟通误差，提升设计效率和质量。通过自动化生成设计文档，确保硬件设计的规范性和一致性，降低设计风险。

模拟电路设计拓展

拓展LLMs在模拟电路（如ADC设计）中的应用，解决当前研究空白。例如，LLMs通过生成SPICE网表或版图描述，推动模拟电路设计的创新和发展。探索LLMs在模拟电路设计中的应用潜力，为模拟电路设计提供新的思路和方法。



综合优化

开发LLMs驱动的综合工具，优化PPA指标。例如，LLMs理解工艺库特性，生成面积更小、功耗更低的布局方案，提升硬件设计的性能和竞争力。结合LLMs的生成能力和综合工具的优化能力，实现硬件设计的高效优化和实现。

EDA集成

开发LLMs驱动的自动化工具链（如ChatEDA），覆盖RTL到GDSII的全流程。例如，自动化生成测试脚本、优化布局布线，缩短设计周期，提升硬件设计的效率和质量。将LLMs与EDA工具链深度融合，实现硬件设计的全流程自动化和智能化。

安全增强

构建抗越狱模型和硬件漏洞数据集，提升生成代码的可靠性。例如，开发基于硬件CWE微调的LLMs，构建大规模硬件漏洞数据集，提升模型对硬件安全漏洞的识别和修复能力。

中国科学院大学
University of Chinese Academy of Sciences

感谢聆听！

赵地、齐洪钢

中国科学院大学