



大语言模型时代下 硬件形式化验证的机遇和挑战

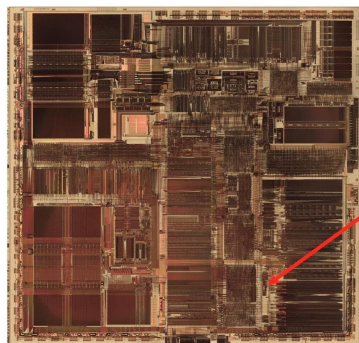
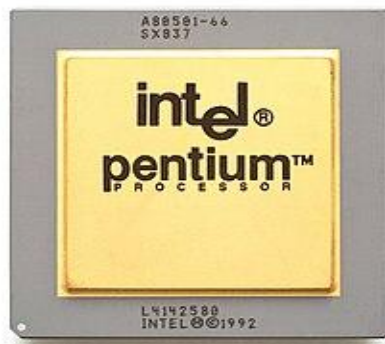
李旻

2025.07.12

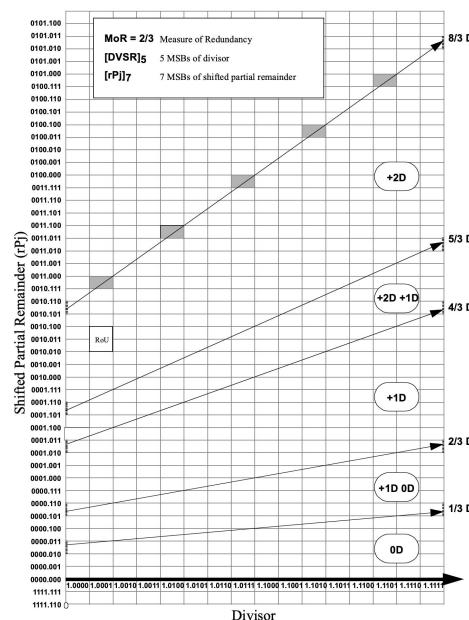
一、讨论范围和形式化验证工具背景



为什么需要验证？



FDIV
bug



浮点除法查找表5/1066 Cells 出错，
九十亿分之一出错概率=>4.75亿美元损失

Back at the time, using 3DS Max for rendering broadcasting animations, I would run into the problem on a weekly basis. Animation rendering batches would fail at a specific frame which would cause the Pentium to freeze. The frame could not be rendered at all, so I would instead render the particular frame on a 486 while the Pentium skipped it and carried on with subsequent frames.

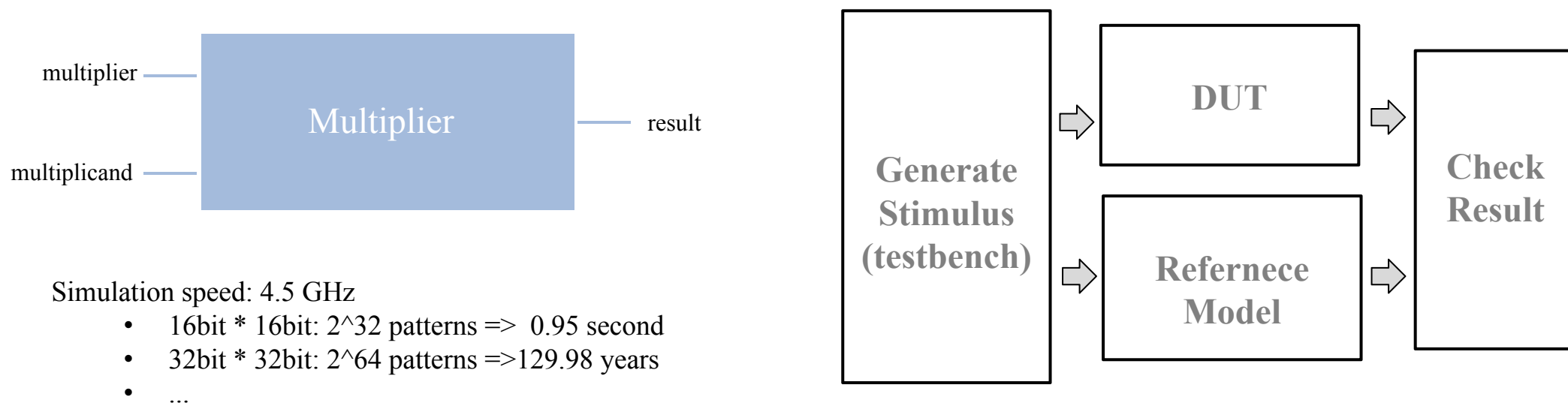
Raytracing was particularly problematic. It wasn't a rare error at all in my experience.

用户反馈当时使用该处理器进行动画渲染，某一帧一定会出错！

算力芯片单Die规格提升
极致PPA提升
极致芯片设计效率提升
=> Bug可能性越高



验证范式：Simulation/Emulation？



对于复杂Datapath电路，基于穷举的Simulation/emulation是不现实的

Intel 08年开始对CPU微架构主要使用Formal；

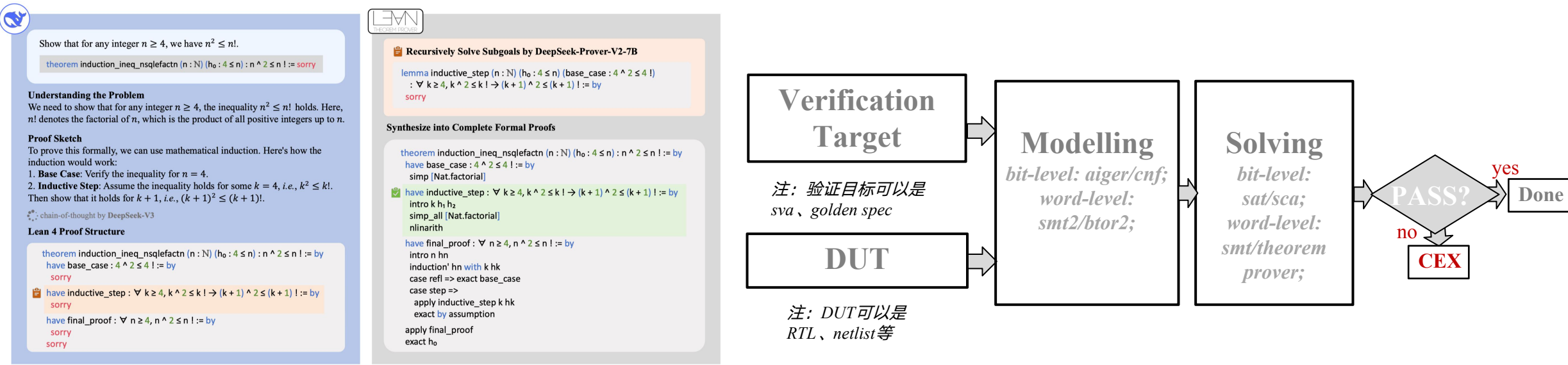
Huawei 18年开始对CPU ALU算子只使用Formal进行验证

Simulation/Emulation

一、讨论范围和形式化验证工具背景



什么是形式化验证？



高数推导题 (DeepSeek-Prover-V2)

硬件形式化验证

- 基于(condition/subject to): $n \geq 4$ / DUT
- 证明 (prove): $n^2 \leq n!$ / dut.out != 0
- 求解器 (solver): 定理证明器lean4 / SAT solver

将硬件验证问题**建模 (Modelling)**成数学上的形式化表示，并调用求解器进行**求解 (Solving)**

一、讨论范围和形式化验证工具背景



硬件形式化验证工具一览（基于求解器技术）

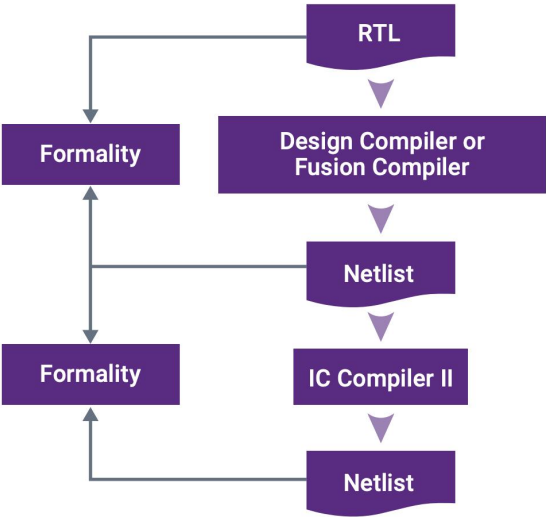
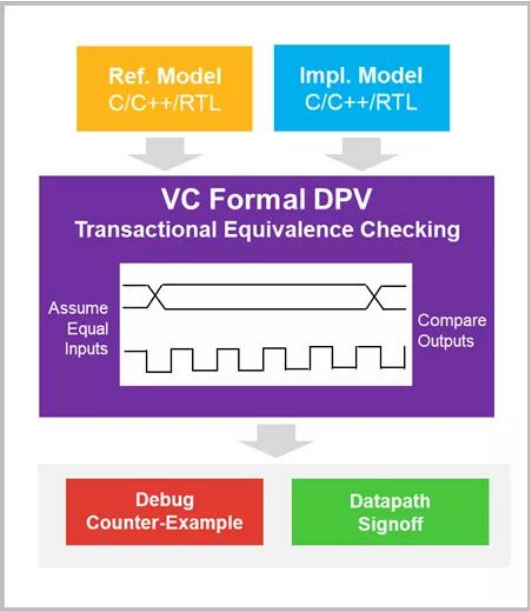
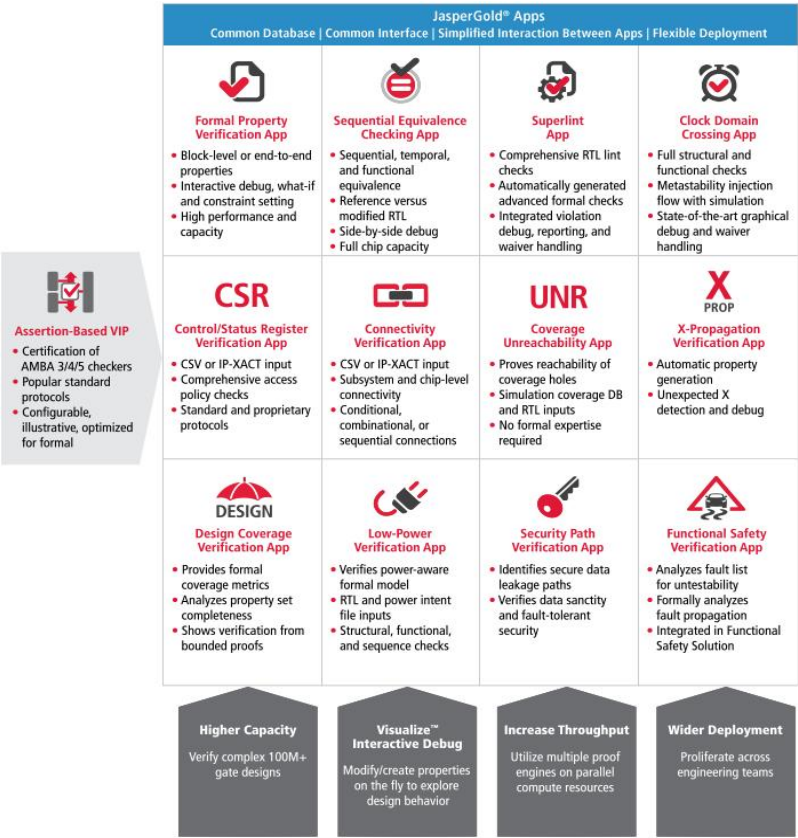


Figure 1: Formality equivalence checking solution

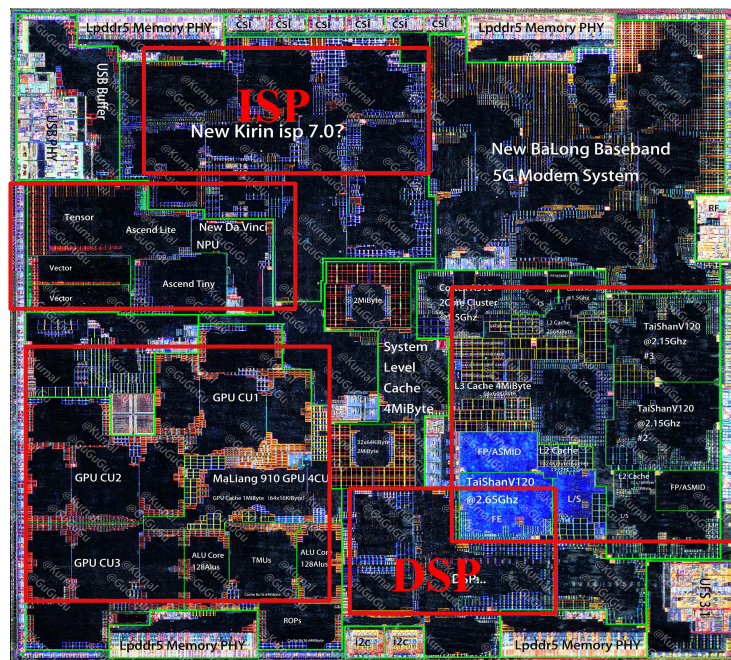
Model Checking

SAT/SMT

一、讨论范围和形式化验证工具背景



AI, Graphics, Processor Designs价值最高的Formal工具 -> Datapath Formal验证



达芬奇
NPU

马良
GPU

泰山
CPU

麒麟9020 SoC

Synopsys Delivers 100X Faster Formal Verification Closure for AI, Graphics, and Processor Designs

VC Formal Datapath Validation Application Enables Broad Market Adoption of HECTOR Technology

Share:

🔗 📄 📧 📧 📧 📧

MOUNTAIN VIEW, Calif., May 23, 2019 /PRNewswire/ --

Highlights:

- VC Formal Datapath Validation application delivers over 100X speed-up in formal verification between a reference C/C++ algorithm and RTL design implementation over conventional techniques
- The new app integrates VC Formal's debug and usability features enabled through Verdi with proven HECTOR technology

<https://news.synopsys.com/2019-05-23-Synopsys-Delivers-100X-Faster-Formal-Verification-Closure-for-AI-Graphics-and-Processor-Designs>

①Synopsys Formal



<https://www.x-epic.com/index.html#/zh/business>

③芯华章 Formal

华大九天、海思等亦有对标工具

Jasper Formal Verification Platform

Smart formal verification apps developed for C/C++ and RTL level verification to find and fix bugs early in the design cycle

OVERVIEW

Find More Bugs in Less Time, Earlier in the Design Process

The Cadence Jasper Formal Verification Platform consists of formal verification apps at the C/C++ and RTL level. They use smart proof technology and machine learning to find and fix bugs and improve verification productivity early in the design cycle.

https://www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification.html

②Cadence Formal

芯天成形式验证平台

EsseFormal

芯天成全功能形式验证工具平台，包含C-to-RTL/RTL-to-Netlist等价验证工具、属性验证工具，以及各种实用验证Apps，贯穿于数字IC设计的各个阶段，为芯片设计过程中的每个环节提供验证解决方案。该平台具有定制化和集成化两大特点，能够精准满足客户需求，大幅降低用户验证时间、提高验证完整性和准确性。

<https://www.gwxeda.com/product/2.html>

④国微芯 Formal

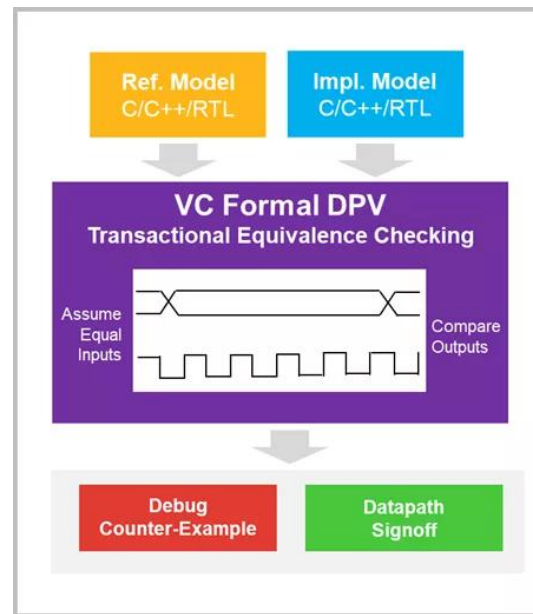
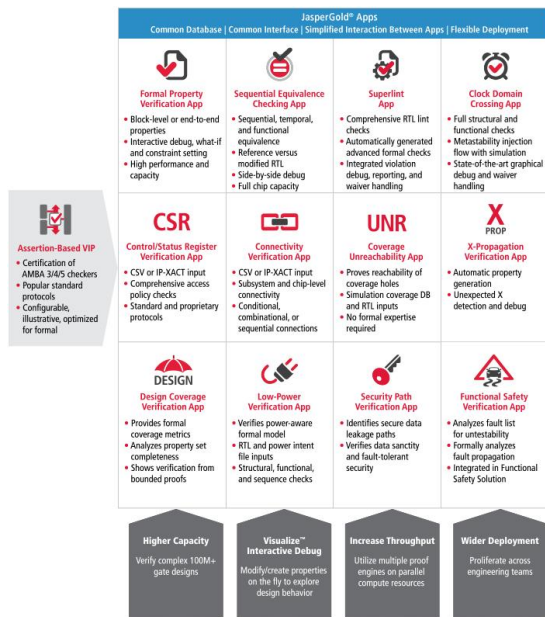
xPU部门/DSP部门/ISP部门都使用
(注：国内头部互联网公司亦在采购)

硬件形式化验证中对算力芯片Signoff
价值最高、同时也是License最贵的工具

一、讨论范围和形式化验证工具背景



Formal验证输入输出件格式由三大家主导



Model checking工具：Cadence JasperGold

- 芯华章GalaxFV
- 华大九天FormalMC
- 国微芯EsseFPV
- ...

Datapath验证工具：Synopsys Hector

- 芯华章HEC
- 华大九天AveCEC
- 国微芯 EsseFECT
- ...

国内EDA厂商仍为跟随发展为主



Formal Property Verification: A case study

```
// address generator/counter
module addr_gen
#( parameter MAX_DATA=16
) ( input en, clk, rst,
  output reg [3:0] addr
);
  initial addr <= 0;

  // async reset
  // increment address when enabled
  always @(posedge clk or posedge rst)
    if (rst)
      addr <= 0;
    else if (en) begin
      if (addr == MAX_DATA-1)
        addr <= 0;
      else
        addr <= addr + 1;
    end
endmodule
```

异步FIFO（节选）

In order to verify our design we must first define properties that it must satisfy. For example, there must never be more than there is memory available. By assigning a signal to count the number of values in the buffer, we can make the following assertion in the code:

```
a_oflow: assert (count <= MAX_DATA);
```

It is also possible to use the prior value of a signal for comparison. This can be used, for example, to ensure that the count is only able to increase or decrease by 1. A case must be added to handle resetting the count directly to 0, as well as if the count does not change. This can be seen in the following code; at least one of these conditions must be true at all times if our design is to be correct.

```
a_counts: assert (count == 0
|| count == $past(count)
|| count == $past(count) + 1
|| count == $past(count) - 1);
```

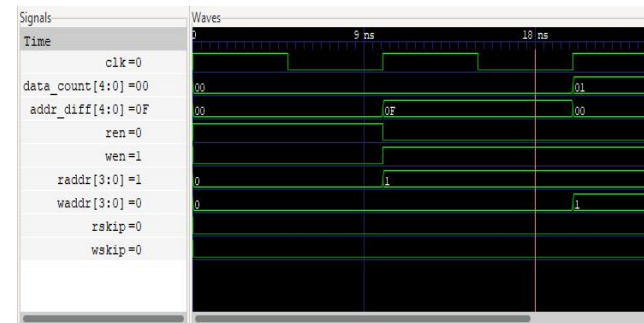
As our count signal is used independently of the read and write pointers, we must verify that the count is always correct. While the write pointer will always be at the same point or *after* the read pointer, the circular buffer means that the write *address* could wrap around and appear *less than* the read address. So we must first perform some simple arithmetic to find the absolute difference in addresses, and then compare with the count signal.

```
assign addr_diff = waddr >= raddr
? waddr - raddr
: waddr + MAX_DATA - raddr;
```

```
a_count_diff: assert (count == addr_diff
|| count == MAX_DATA && addr_diff == 0);
```

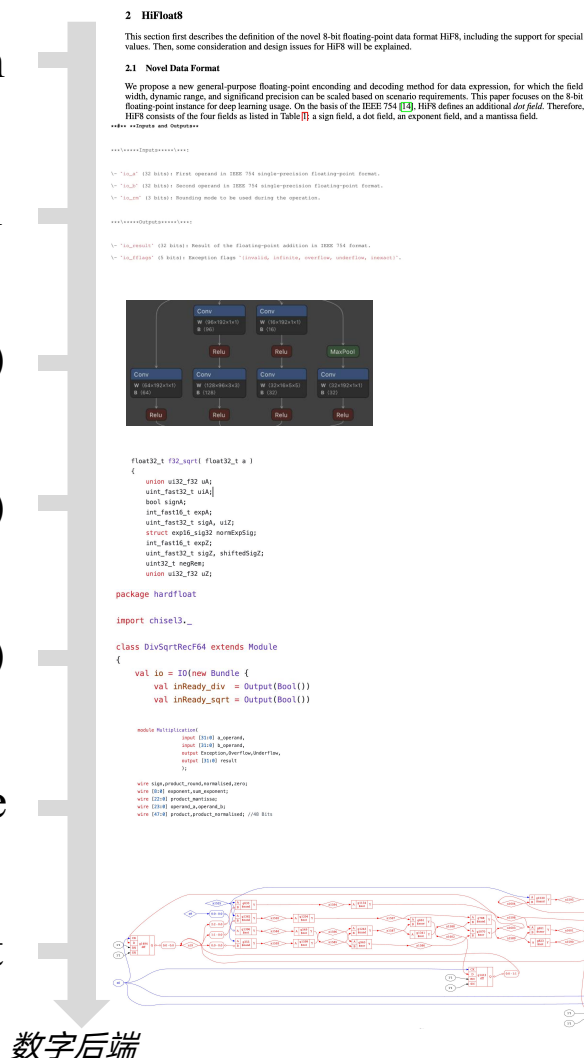
Formal asserts

ref: <https://yosyshq.readthedocs.io/projects/sby/en/latest/quickstart.html#first-in-first-out-fifo-buffer>



Counter Example

当前成功案例



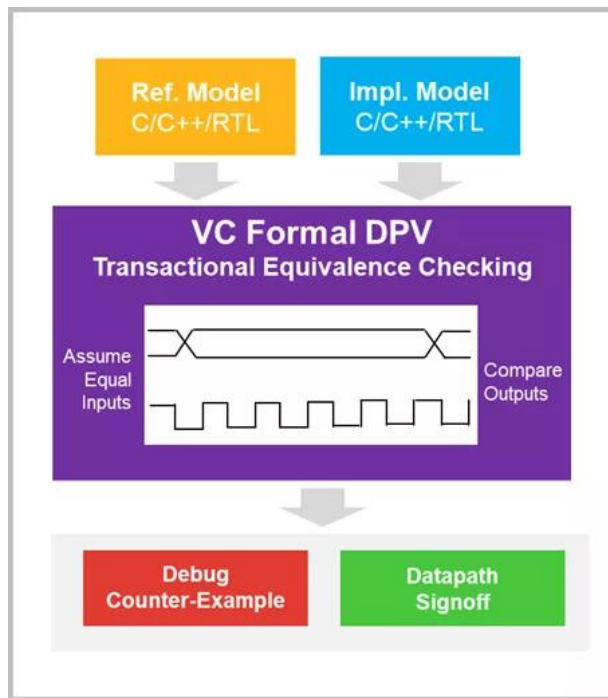
更多可能

LLM可以改变当前瀑布式设计流程
=>设计左移

二、新设计范式=>新验证工具



数字前端电路跨模态等价性检查只涉及2种场景



以Synopsys Hector/Cadence C2RTL为代表

C-RTL:
Transaction Equivalence Checking

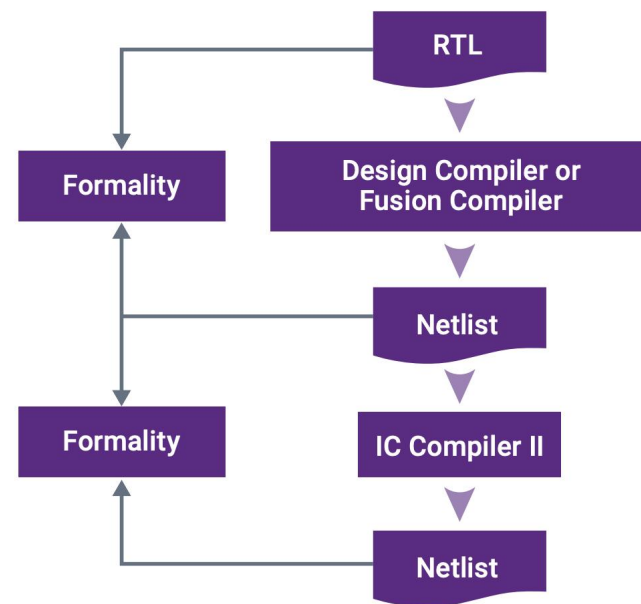


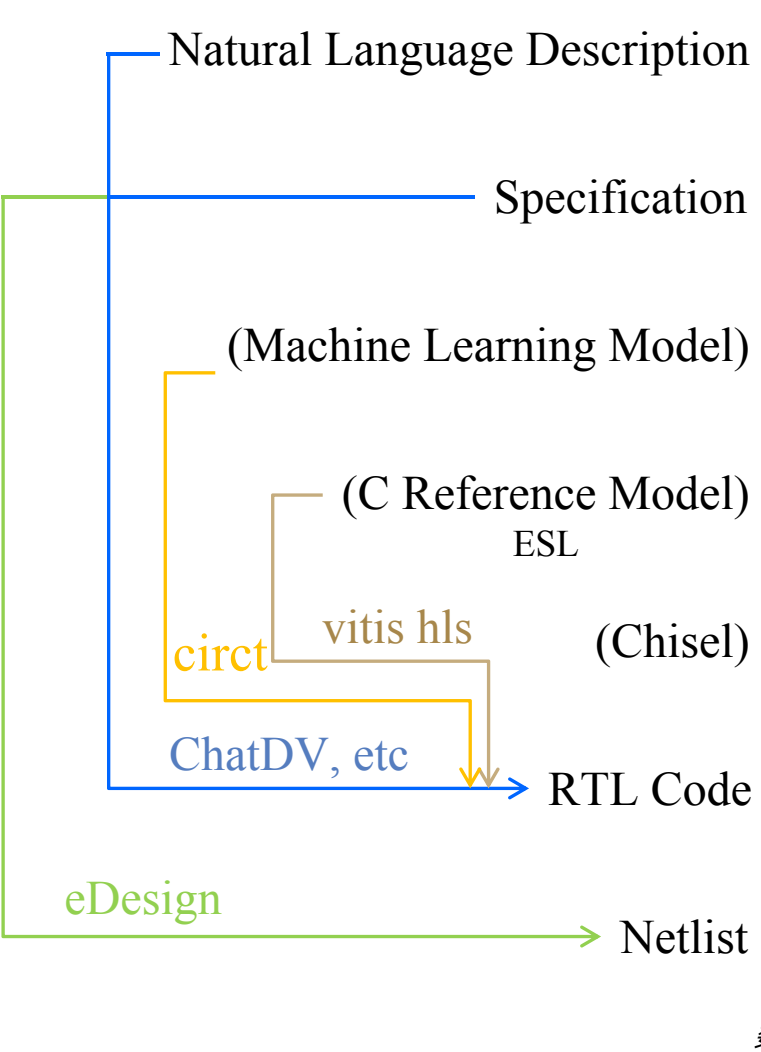
Figure 1: Formality equivalence checking solution

以Synopsys Formality/Cadence Conformal为代表

RTL-Netlist:
Combinatorial Equivalence Checking



验证左移



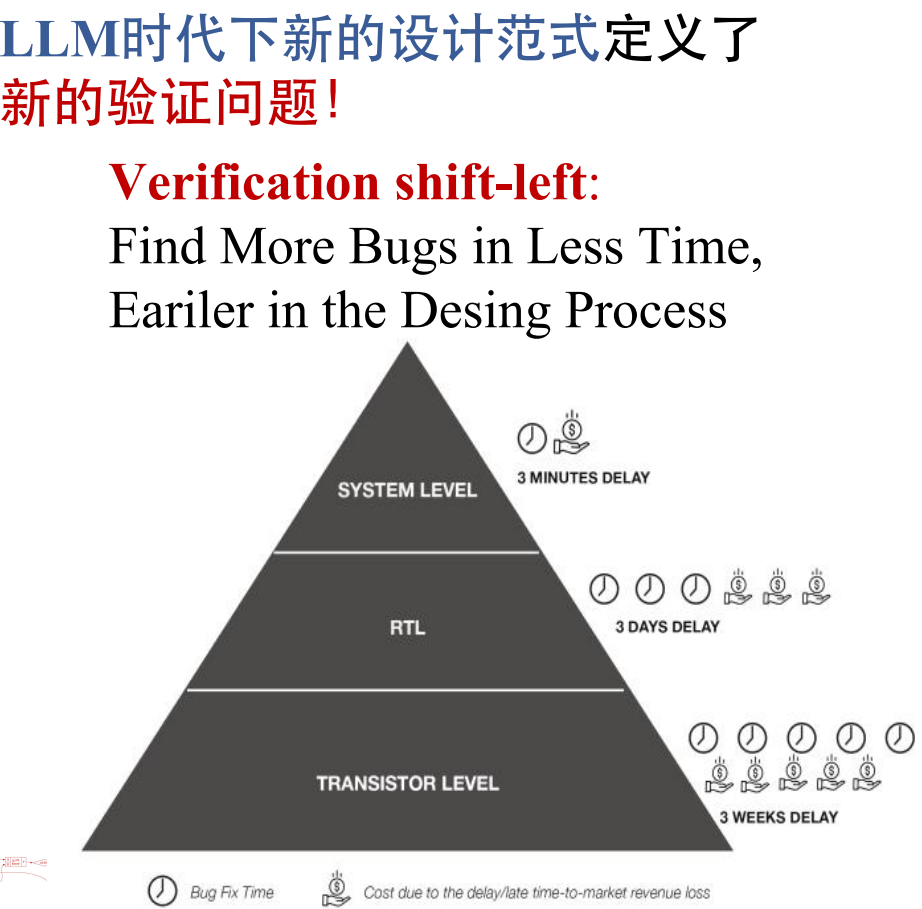
2 HIFloat8

This section first describes the definition of the novel 8-bit floating-point data format HIF8, including the support for special values. Then, some consideration and design issues for HIF8 will be explained.

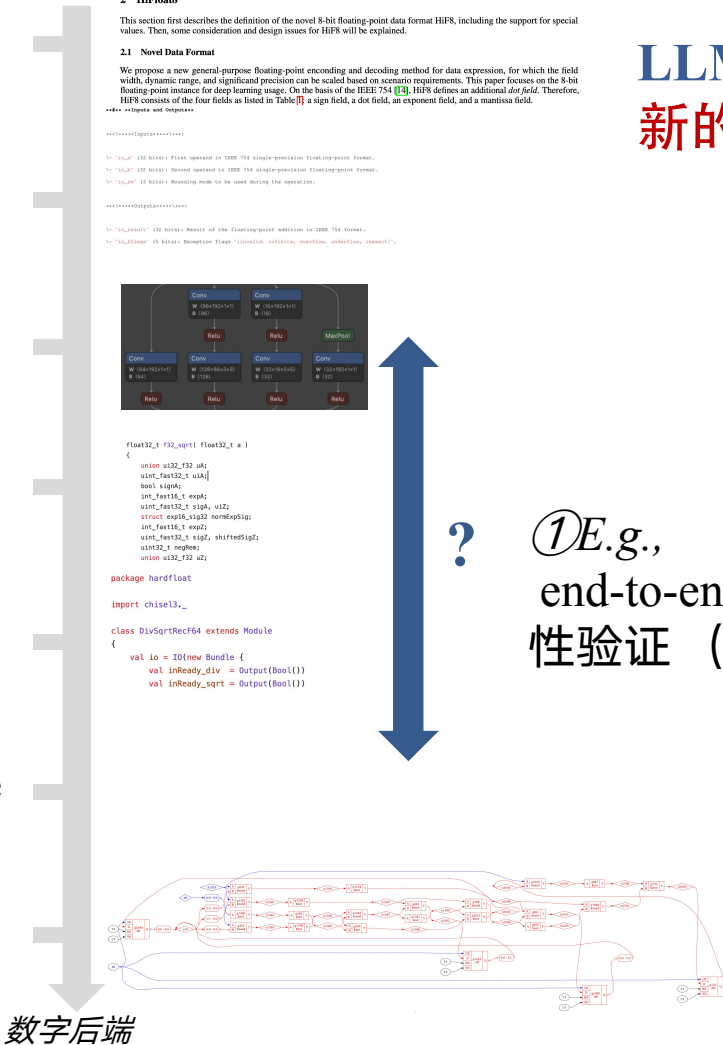
2.1 Novel Data Format

We propose a new general-purpose floating-point encoding and decoding method for data expression, for which the field width, dynamic range, and significant precision can be scaled based on scenario requirements. This paper focuses on the 8-bit floating-point instance for deep learning usage. On the basis of the IEEE 754 [14], HIF8 defines an additional *dot* field. Therefore, HIF8 consists of the four fields as listed in Table 3: a sign field, a dot field, an exponent field, and a mantissa field.

code snippets and diagrams



验证左移

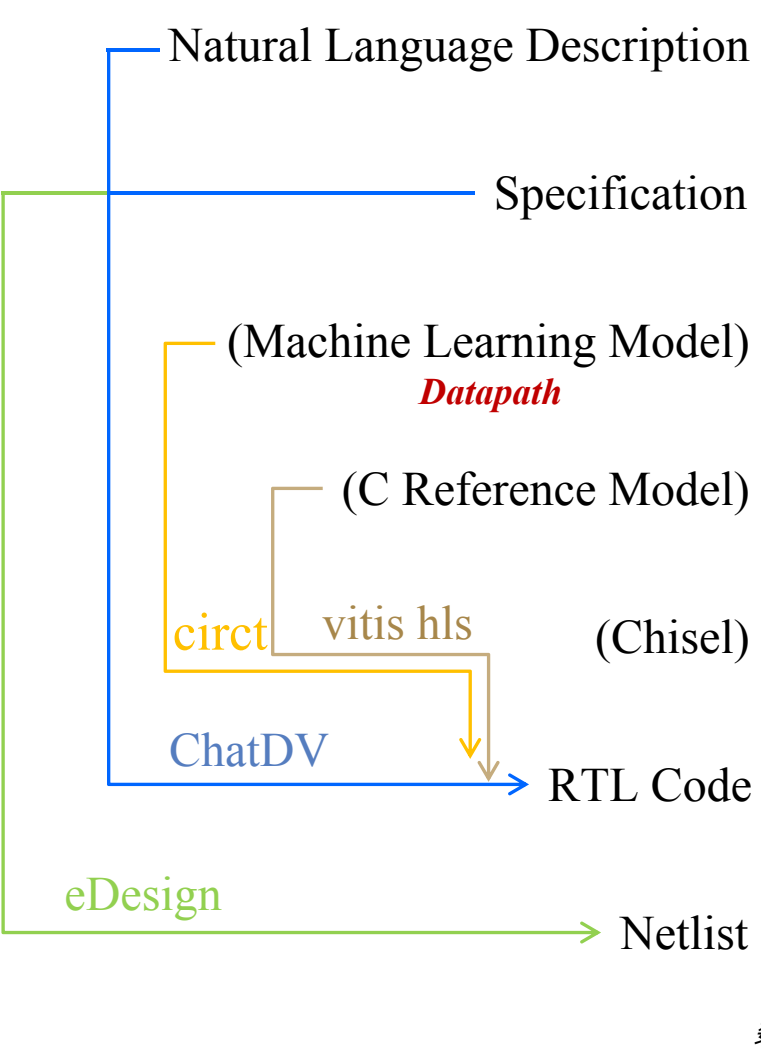


LLM时代下新的设计范式定义了新的验证问题！

①E.g.,
end-to-end ISP CNN模型和Verilog进行等价性验证 （客户真实需求）



验证左移



2 HIFloat8

This section first describes the definition of the novel 8-bit floating-point data format HIF8, including the support for special values. Then, some consideration and design issues for HIF8 will be explained.

2.1 Novel Data Format

We propose a new general-purpose floating-point encoding and decoding method for data expression, for which the field width, dynamic range, and significant precision can be scaled based on scenario requirements. This paper focuses on the 8-bit floating-point instance for deep learning usage. On the basis of the IEEE 754 [14], HIF8 defines an additional *dot field*. Therefore, HIF8 consists of the four fields as listed in Table 3: a sign field, a dot field, an exponent field, and a mantissa field.

code snippets and datapaths

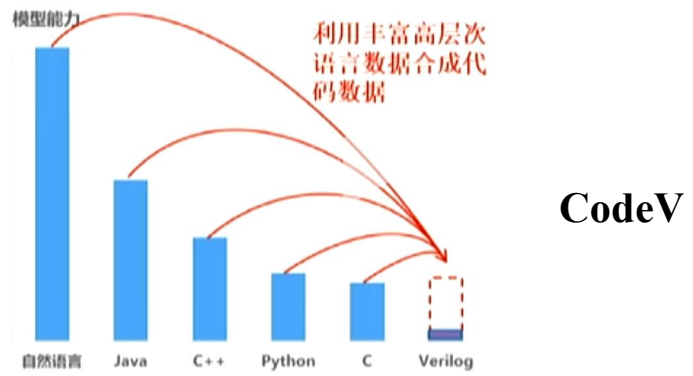
```
float32_t f32_sqrt(float32_t a)
{
    union u32_f32 u;
    u32_f32_t u32;
    bool sign;
    int_fast16_t exp;
    u32_f32_t sig;
    struct exp8_sig8 normExpSig;
    int_fast16_t exp2;
    u32_f32_t sig2, shiftExpSig;
    u32_f32_t expres;
    union u32_f32 u2;

    package hardfloat

    import chisel3._

    class DivSqrtRecF64 extends Module
    {
        val io = IO(new Bundle {
            val inReady_div = Output(Bool())
            val inReady_sqrt = Output(Bool())
        })
    }
```

LLM时代下新的设计范式定义了新的验证问题!



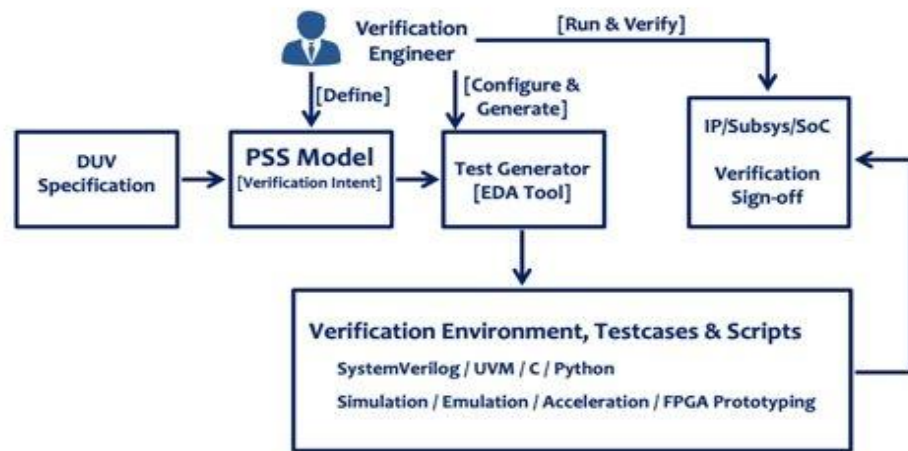
Code Agent

=?

- ②E.g.,
- 通过Code agent从Spec生成Chisel, 功能:(
 - 通过Code agent从Spec生成C RM, 功能:)
 - 通过Chisel和CRM功能对齐, 来保证Chisel的正确性
- =>Chisel和C RM的等价性验证



思考1. Formal门槛和成本如何降低？

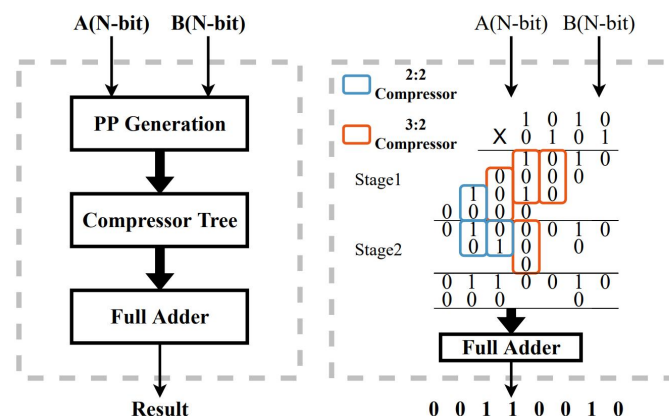


Formal验证上手门槛高

- 对体系架构、各类协议、验证策略、形式验证方法学（硬件抽象能力）、SVA等技能均需掌握
- 4000+验证工程师，只有200+人会使用Formal工具

Abstraction Techniques for Complex DUT:

- 1) Case splitting
- 2) Cut-point/black box
- 3) Assume guarantee



Typical time cost

- FP Add: 0.5hour - 2days
- FP Mul: 2days - 2 weeks
- FP Div: 1 - 6 months

能用≠会用

- 验证工程师写的Formal环境质量层次不齐，同样的case不同的环境，Formal收敛cost可能有一个数量级以上差别



思考1. Formal门槛和成本如何降低？

Time Cost for C2RTL verification (setup to fully proven)

- TCL refinement loop to run correctly
 - fix constraints to run correctly
 - fix C&RTL input/output mapping
- TCL refinement loop to convergence
 - try case splitting, rerun, might still inconclusive. however we find the bottleneck ...
 - to prove the bottleneck, try A, inconclusive
 - to prove the bottleneck, try B, inconclusive
 - ...
 - proved a tiny step
 - ...
 - proved all properties
- Rerun the final worked TCL

现有商用Formal工具很难解决Scalability，只能尽可能提供多的手段，供验证人员循环迭代验证环境，构造诸多tcl脚本

Industrial Examples:
xxx用例: 2000+ tcl验证环境,
花一年才达到完备收敛

ref:
https://www.bilibili.com/video/BV1fJ23YQE9Y/?spm_id_from=333.1387.favlist.content.click&vd_source=637e596cabfb504f6a386a96f7ec263

Show that for any integer $n \geq 4$, we have $n^2 \leq n!$.

```
theorem induction_ineq_nsqlefactn (n : N) (h0 : 4 ≤ n) : n ^ 2 ≤ n ! := sorry
```

Understanding the Problem
We need to show that for any integer $n \geq 4$, the inequality $n^2 \leq n!$ holds. Here, $n!$ denotes the factorial of n , which is the product of all positive integers up to n .

Proof Sketch
To prove this formally, we can use mathematical induction. Here's how the induction would work:
1. **Base Case:** Verify the inequality for $n = 4$.
2. **Inductive Step:** Assume the inequality holds for some $k = 4$, i.e., $k^2 \leq k!$. Then show that it holds for $k + 1$, i.e., $(k + 1)^2 \leq (k + 1)!$.

chain-of-thought by DeepSeek-V3

Lean 4 Proof Structure

```
theorem induction_ineq_nsqlefactn (n : N) (h0 : 4 ≤ n) : n ^ 2 ≤ n ! := by
  have base_case : 4 ^ 2 ≤ 4 ! := by
    sorry
  have inductive_step : ∀ k ≥ 4, k ^ 2 ≤ k ! → (k + 1) ^ 2 ≤ (k + 1) ! := by
    sorry
  have final_proof : ∀ n ≥ 4, n ^ 2 ≤ n ! := by
    sorry
  sorry
```

Recursively Solve Subgoals by DeepSeek-Prover-V2-7B

```
lemma inductive_step (n : N) (h0 : 4 ≤ n) (base_case : 4 ^ 2 ≤ 4 !)  
  : ∀ k ≥ 4, k ^ 2 ≤ k ! → (k + 1) ^ 2 ≤ (k + 1) ! := by  
  sorry
```

Synthesize into Complete Formal Proofs

```
theorem induction_ineq_nsqlefactn (n : N) (h0 : 4 ≤ n) : n ^ 2 ≤ n ! := by
  have base_case : 4 ^ 2 ≤ 4 ! := by
    simp [Nat.factorial]
  have inductive_step : ∀ k ≥ 4, k ^ 2 ≤ k ! → (k + 1) ^ 2 ≤ (k + 1) ! := by
    intro k h1 h2
    simp_all [Nat.factorial]
    nlinarith
  have final_proof : ∀ n ≥ 4, n ^ 2 ≤ n ! := by
    intro n hn
    induction' hn with k hk
    case refl => exact base_case
    case step =>
      apply inductive_step k hk
      exact by assumption
  apply final_proof
  exact h0
```

1. Decompose a theorem into sub-goals

2. Solving Sub-goals recursively with LLM

版本	关键技术	核心能力提升	数学层级覆盖
V1	大规模合成数据	解决数据稀缺，实现 FIMO 全验证	高中竞赛
V1.5	截断-恢复机制 + RMaxTS + RLPAF	动态纠错、树搜索多样化探索	高中→本科
V2	子目标分解 + 递归证明	非形式/形式推理统一、复杂问题分层破解	本科→研究生

DeepSeek-Prover

LLM Prover实现复杂Datapath电路自动证明？
实现LLM对复杂验证目标的自动抽象和拆解，降低Formal工具使用门槛，进一步提高收敛效率

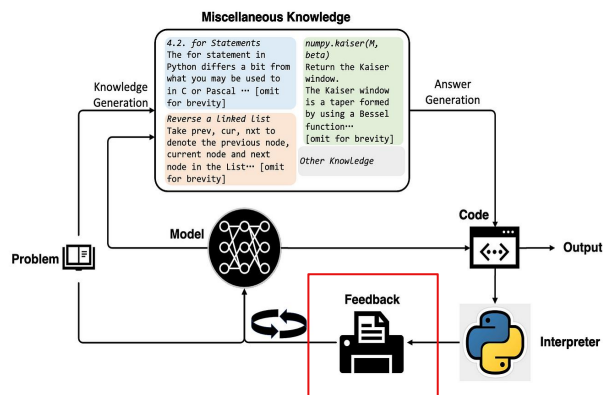
难点：Scalability



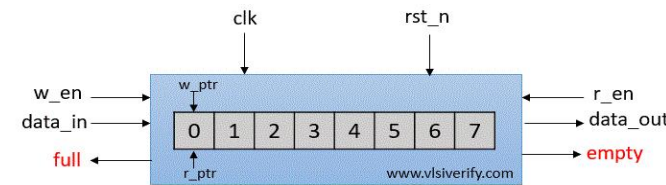
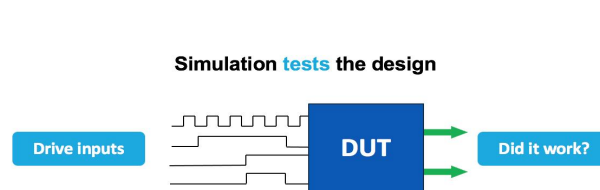
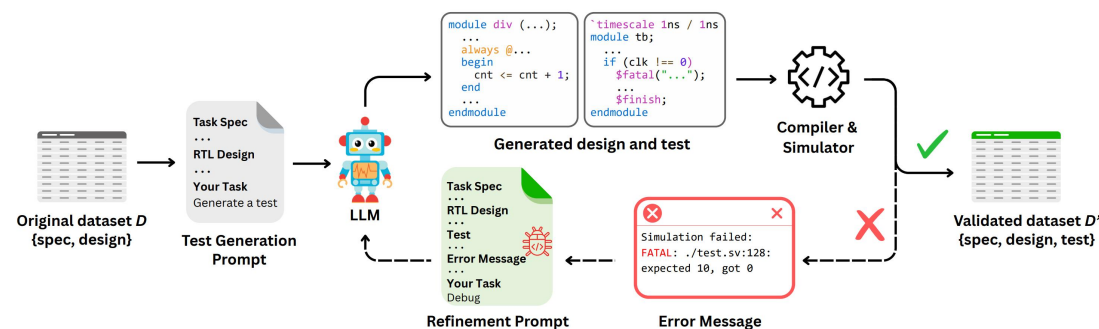
思考2. 形式化验证如何反哺设计？



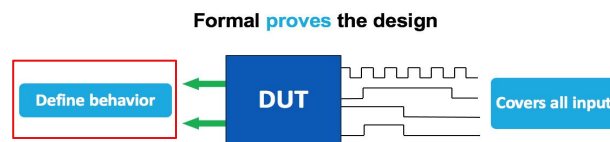
Gemini Cli



工具Log作为Agent在当前迭代得到的反馈，知道Agent在下次迭代**Self-debugging**



Synchronous FIFO



E.g., 异步FIFO
Simulation => hundreds of testbench
Formal => 7 SVA properties

Formal Env.是设计意图的形式化抽象，应该无损表示所有的设计意图；Golden

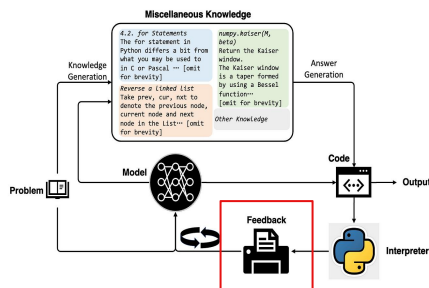
①**Formal收敛** 《=》设计正确

②**Formal Feedback迅速/信息量大**

远比Simulation更适合作为“LLM4xPU”迭代反馈的核心



思考2. 形式化验证如何反哺设计？



Assistant #
4 days ago

I understand. This is a baseline v2 test for generating IEEE 754 compliant floating-point multiplication units (FMUL) using Claude Code with complete design freedom and unlimited iterations. The goal is to achieve 100% pass rate (330/330 tests) through unrestricted creative iteration.

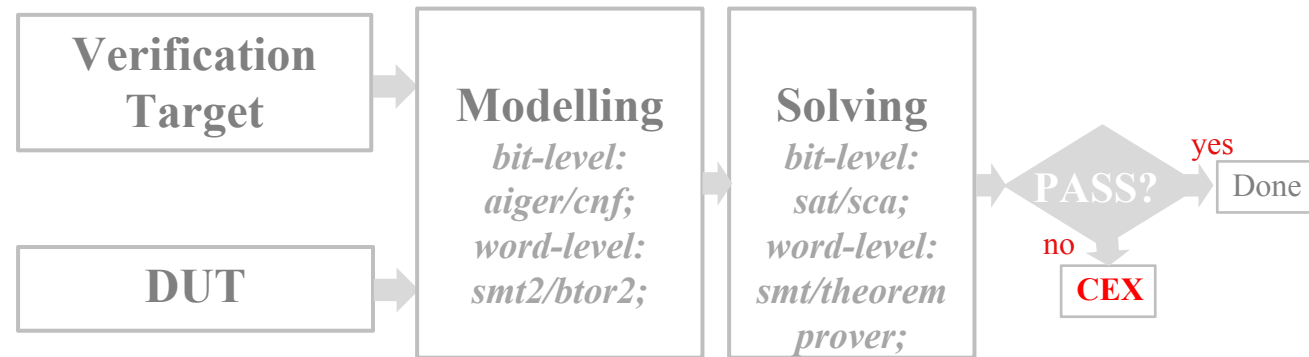
Key aspects:

- **Target:** 32-bit IEEE 754 FMUL with 5 rounding modes
- **Testing:** 330 test vectors (66 combinations x 5 rounding modes)
- **Approach:** Unlimited iterations with complete design freedom
- **Interface:** Standard module with io_a, io_b, io_rm inputs and io_result, io_flags outputs

Ready to begin unrestricted FMUL generation.

成功案例：
FPU Agent using Claude Code and Simulation
基于testfloat生成600+ testbench

FMUL: 24 iterations, ~2 hrs
FADD: 17 iterations, ~1 hr
Compare: 1 iterations
INT2FP/FP2INT: 18 iterations



Formal工具流程包含了：

- 1) Lint
- 2) PASS/FAIL
- 3) Fail with counter-example for simulation/debugging
- 4) 自研EquivFusion，可以拿到工具内任何信息；E.g., COI, Line Cov.

Formal作为“RTL生成”迭代反馈的核心

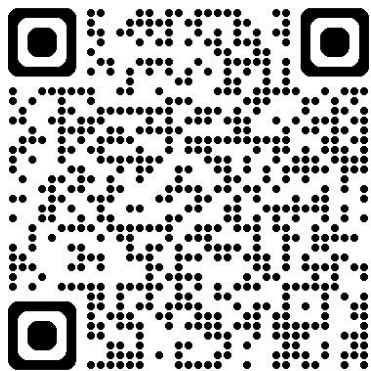
Formal + Chip Design Agent: 利用Formal提高 self-debugging上限
从而提高Chip Design Agent设计效率

难点：Scalability

欢迎交流!

On-going project: **EquivFusion**

Website: <https://formind.netlify.app/>



WeChat



扫一扫上面的二维码图案，加我为朋友。