東南大學 SCHOOL OF INTEGRATED CIRCUITS, SEU
集成电路学院

形式化方法在硬件验证软件中的应用和展望

李旻

东南大学 集成电路学院

2025.08.15

个人介绍

李旻

- ■ 基本情况
  - 职务　　东南大学集成电路学院　　上岗研究员
  - 研究方向　　硬件形式化验证、智能EDA

- ■ 工作 & 教育经历

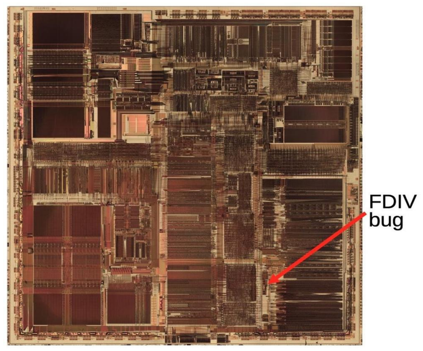
2025.07-现在	东南大学/EDA国创中心
2023.05-2025.06	华为诺亚方舟实验室
2018.08-2023.03	香港中文大学（导师：**徐强**教授）

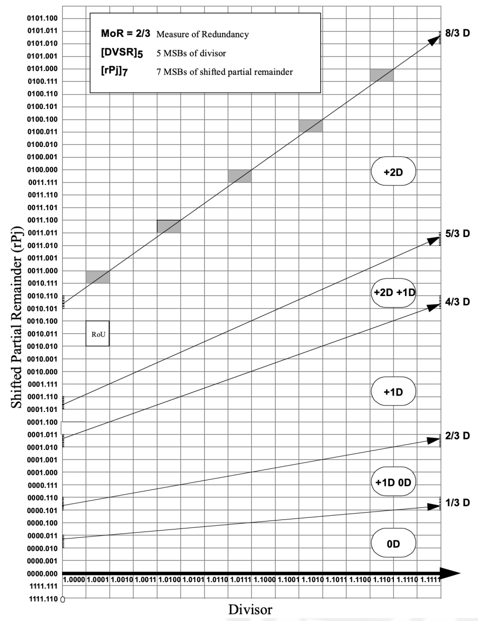
- ■ 业绩情况
  - 业界：参与海思国产自研形式化工具开发，帮助华为多个处理器芯片验证收敛
  - 论文：近五年发表EDA领域高水平会议20+
  - 科研项目：承担国家优青（海外）、海思EDA、港府多项科研项目

目录

硬件形式化验证工具简介

为什么需要硬件形式化验证?





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 iny 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

硬件形式化验证工具简介

数字电路前端流程

Natural Language Description

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.

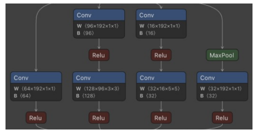
2.1 Novel Data Format

We propose a new general-purpose floating-point enconding and decoding method for data expression, for which the field width, dynamic range, and significand 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 1: a sign field, a dot field, an exponent field, and a mantissa field.

Specification

```
**#** **Inputs and Outputs**

***\*****Inputs*****\***:

\- `io_a` (32 bits): First operand in IEEE 754 single-precision floating-point format.

\- `io_b` (32 bits): Second operand in IEEE 754 single-precision floating-point format.

\- `io_rm` (3 bits): Rounding mode to be used during the operation.

***\*****Outputs*****\***:

\- `io_result` (32 bits): Result of the floating-point addition in IEEE 754 format.

\- `io_fflags` (5 bits): Exception flags `{invalid, infinite, overflow, underflow, inexact}`.
```

(Machine Learning Model)



(C Reference Model)
ESL

```
float32_t f32_sqrt( float32_t a )
{
    union ui32_f32 uA;
    uint_fast32_t uiA;
    bool signA;
    int_fast16_t expA;
    uint_fast32_t sigA, uiZ;
    struct exp16_sig32 normExpSig;
    int_fast16_t expZ;
    uint_fast32_t sigZ, shiftedSigZ;
    uint32_t negRem;
    union ui32_f32 uZ;
```

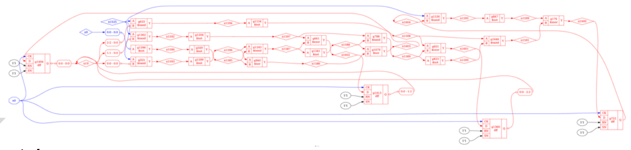
(Chisel)

```
package hardfloat

import chisel3._

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

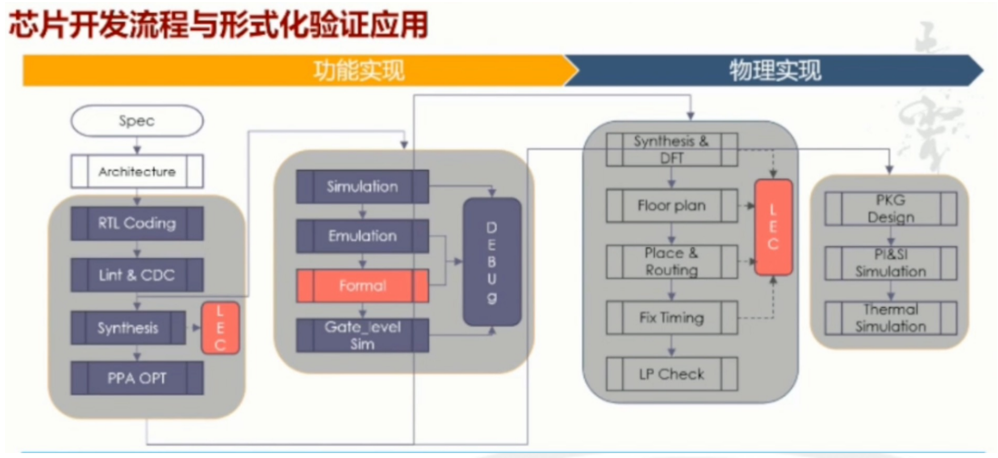
RTL Code

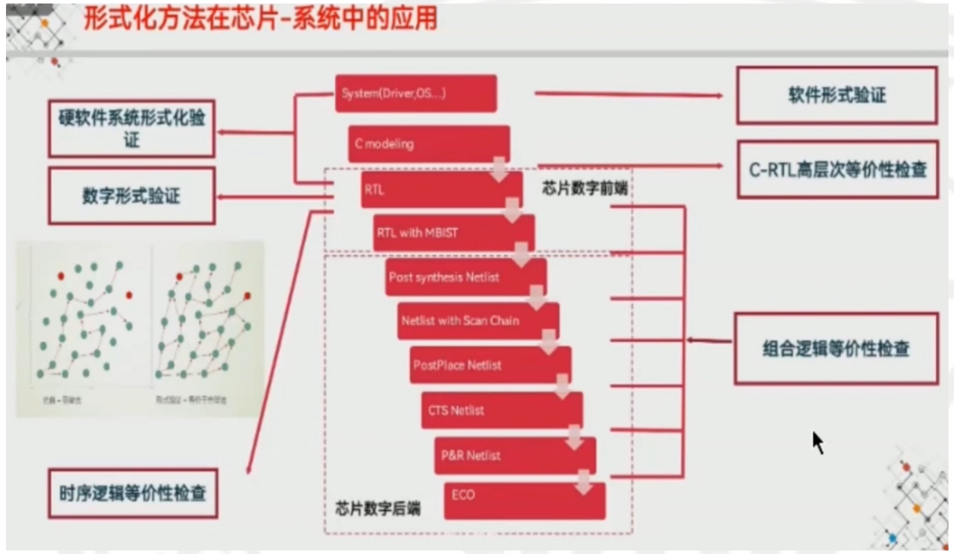
```
module Multiplication(
                input [31:0] a_operand,
                input [31:0] b_operand,
                output Exception,Overflow,Underflow,
                output [31:0] result
                );

wire sign,product_round,normalised,zero;
wire [8:0] exponent,sum_exponent;
wire [22:0] product_mantissa;
wire [23:0] operand_a,operand_b;
wire [47:0] product,product_normalised; //48 Bits
```

Netlist



数字后端

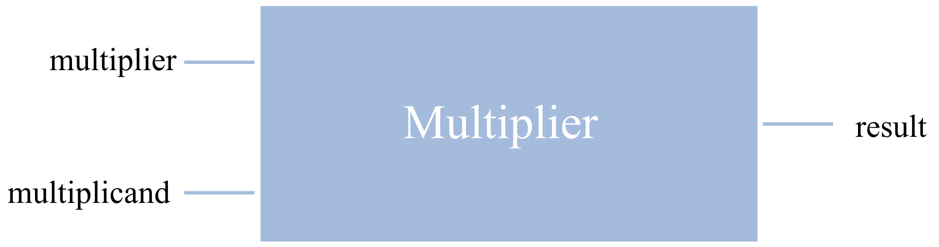




src: 2021 CCF-中国软件大会 海思/阿卡斯

硬件形式化验证工具简介

验证范式：Simulation or Formal?

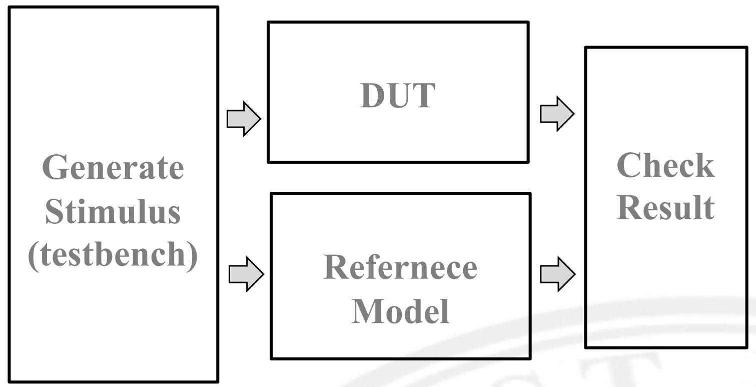


Simulation speed: 4.5 GHz

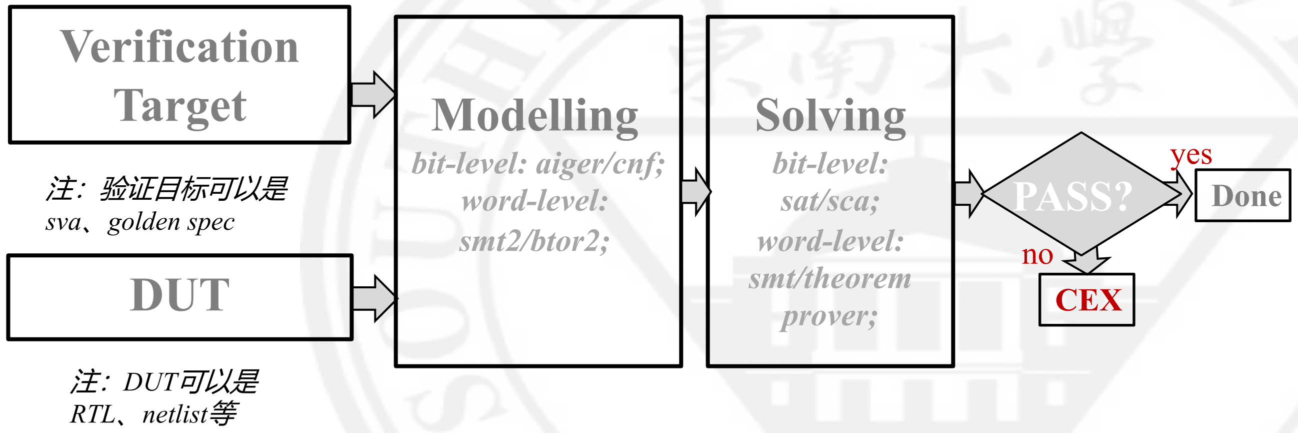
- 16bit * 16bit: 2^32 patterns => 0.95 second
- 32bit * 32bit: 2^64 patterns =>129.98 years
- …

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；



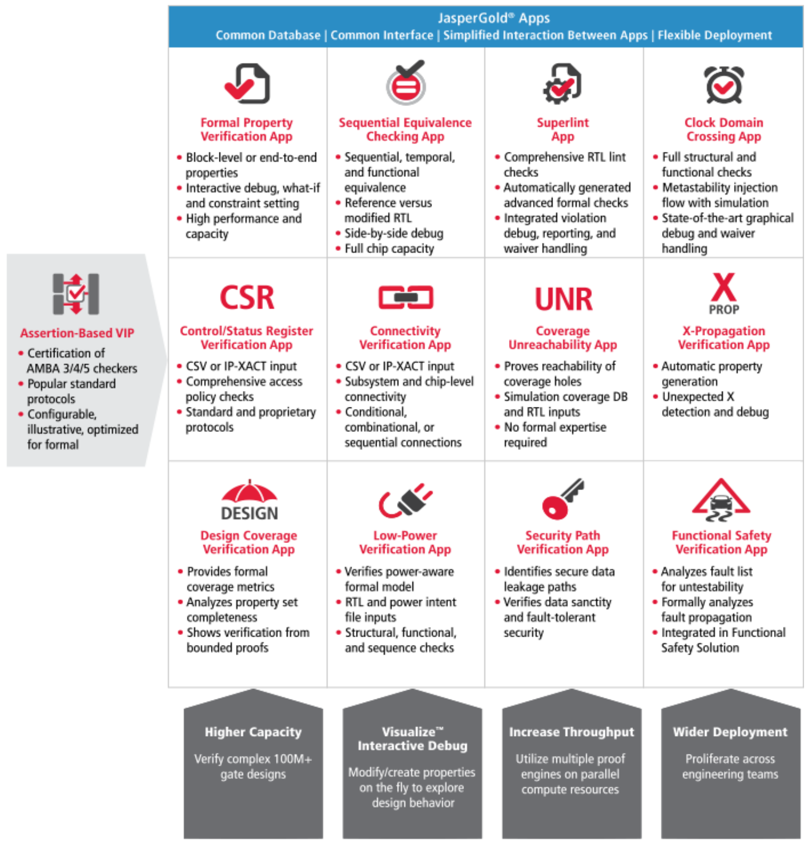
Simulaiton/Emulation



Formal

硬件形式化验证工具简介

商用硬件形式化验证工具一览（基于求解器技术）



JasperGold® Apps

Common Database | Common Interface | Simplified Interaction Between Apps | Flexible Deployment

Formal Property Verification App
- Block-level or end-to-end properties
- Interactive debug, what-if and constraint setting
- High performance and capacity

Sequential Equivalence Checking App
- Sequential, temporal, and functional equivalence
- Reference versus modified RTL
- Side-by-side debug
- Full chip capacity

Superlint App
- Comprehensive RTL lint checks
- Automatically generated advanced formal checks
- Integrated violation debug, reporting, and waiver handling

Clock Domain Crossing App
- Full structural and functional checks
- Metastability injection flow with simulation
- State-of-the-art graphical debug and waiver handling

Assertion-Based VIP
- Certification of AMBA 3/4/5 checkers
- Popular standard protocols
- Configurable, illustrative, optimized for formal

CSR
Control/Status Register Verification App
- CSV or IP-XACT input
- Comprehensive access policy checks
- Standard and proprietary protocols

Connectivity Verification App
- CSV or IP-XACT input
- Subsystem and chip-level connectivity
- Conditional, combinational, or sequential connections

UNR
Coverage Unreachability App
- Proves reachability of coverage holes
- Simulation coverage DB and RTL inputs
- No formal expertise required

X PROP
X-Propagation Verification App
- Automatic property generation
- Unexpected X detection and debug

DESIGN
Design Coverage Verification App
- Provides formal coverage metrics
- Analyzes property set completeness
- Shows verification from bounded proofs

Low-Power Verification App
- Verifies power-aware formal model
- RTL and power intent file inputs
- Structural, functional, and sequence checks

Security Path Verification App
- Identifies secure data leakage paths
- Verifies data sanctity and fault-tolerant security

Functional Safety Verification App
- Analyzes fault list for untestability
- Formally analyzes fault propagation
- Integrated in Functional Safety Solution

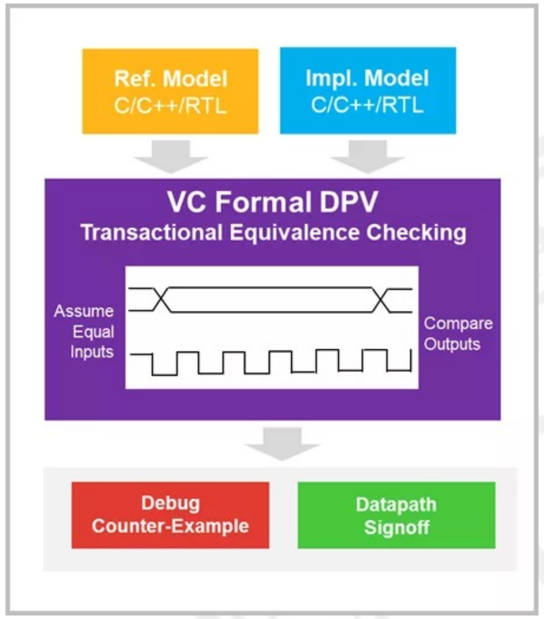
Higher Capacity
Verify complex 100M+ gate designs

Visualize™ Interactive Debug
Modify/create properties on the fly to explore design behavior

Increase Throughput
Utilize multiple proof engines on parallel compute resources

Wider Deployment
Proliferate across engineering teams

Model Checking



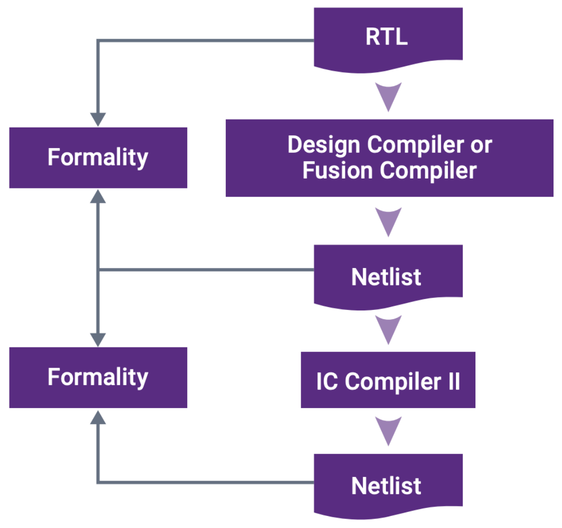
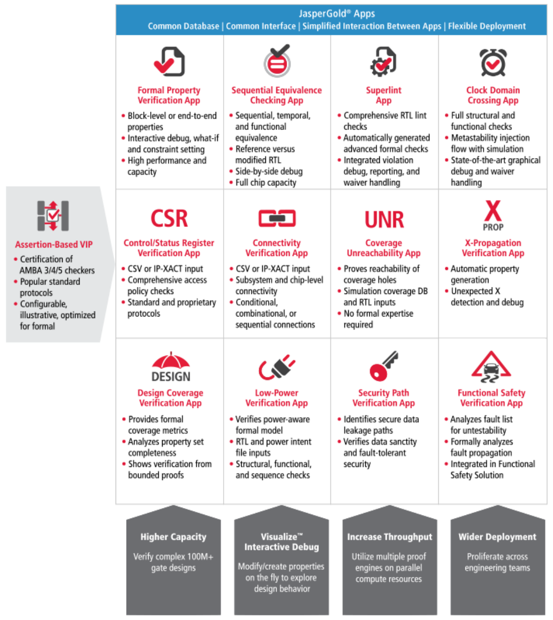


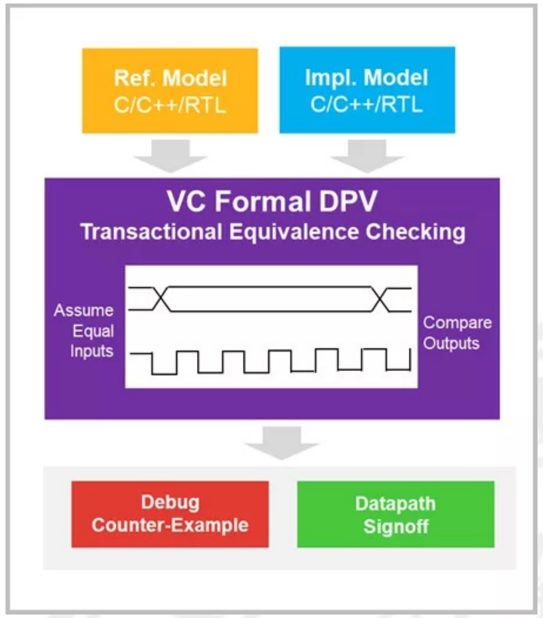
Figure 1: Formality equivalence checking solution

SAT/SMT

硬件形式化验证工具简介

Formal验证输入输出件格式由三大家主导


JasperGold® Apps
Common Database | Common Interface | Simplified Interaction Between Apps | Flexible Deployment
Formal Property Verification App
• Block-level or end-to-end properties
• Interactive debug, what-if and constraint setting
• High performance and capacity
Sequential Equivalence Checking App
• Sequential, temporal, and functional equivalence
• Reference versus modified RTL
• Side-by-side debug
• Full chip capacity
Superlint App
• Comprehensive RTL lint checks
• Automatically generated advanced formal checks
• Integrated violation debug, reporting, and waiver handling
Clock Domain Crossing App
• Full structural and functional checks
• Metastability injection flow with simulation
• State-of-the-art graphical debug and waiver handling
Assertion-Based VIP
• Certification of AMBA 3/4/5 checkers
• Popular standard protocols
• Configurable, illustrative, optimized for formal
CSR
Control/Status Register Verification App
• CSV or IP-XACT input
• Comprehensive access policy checks
• Standard and proprietary protocols
Connectivity Verification App
• CSV or IP-XACT input
• Subsystem and chip-level connectivity
• Conditional, combinational, or sequential connections
UNR
Coverage Unreachability App
• Proves reachability of coverage holes
• Simulation coverage DB and RTL inputs
• No formal expertise required
X PROP
X-Propagation Verification App
• Automatic property generation
• Unexpected X detection and debug
DESIGN
Design Coverage Verification App
• Provides formal coverage metrics
• Analyzes property set completeness
• Shows verification from bounded proofs
Low-Power Verification App
• Verifies power-aware formal model
• RTL and power intent file inputs
• Structural, functional, and sequence checks
Security Path Verification App
• Identifies secure data leakage paths
• Verifies data sanctity and fault-tolerant security
Functional Safety Verification App
• Analyzes fault list for untestability
• Formally analyzes fault propagation
• Integrated in Functional Safety Solution
Higher Capacity
Verify complex 100M+ gate designs
Visualize™ Interactive Debug
Modify/create properties on the fly to explore design behavior
Increase Throughput
Utilize multiple proof engines on parallel compute resources
Wider Deployment
Proliferate across engineering teams



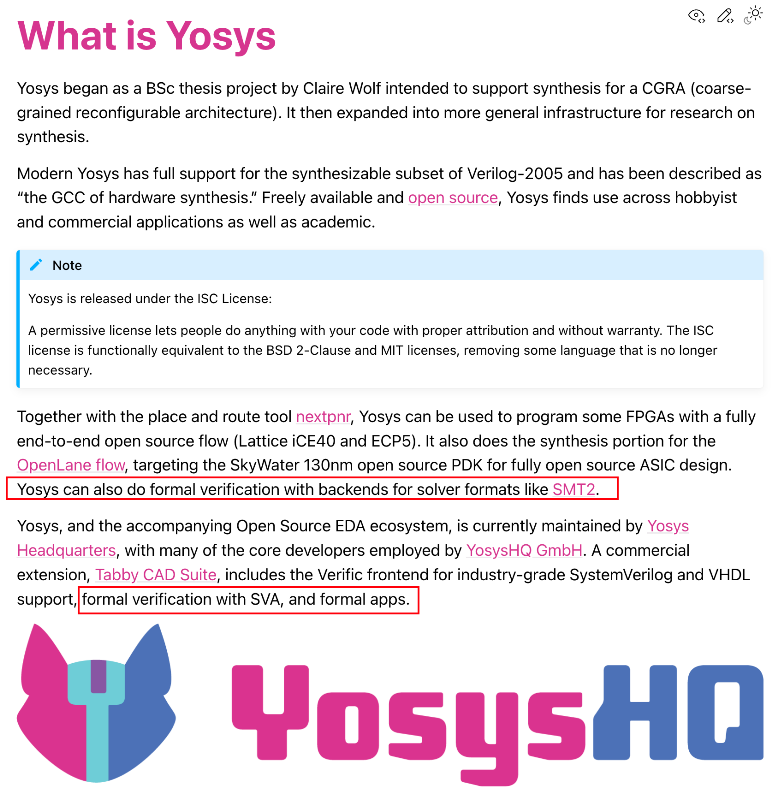
Model checking工具：Cadence JasperGold
- 芯华章GalaxFV
- 华大九天FormalMC
- 国微芯EsseFPV
- …

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

硬件形式化验证工具简介

学术界相关工具：综合/验证



What is Yosys

Yosys began as a BSc thesis project by Claire Wolf intended to support synthesis for a CGRA (coarse-grained reconfigurable architecture). It then expanded into more general infrastructure for research on synthesis.

Modern Yosys has full support for the synthesizable subset of Verilog-2005 and has been described as "the GCC of hardware synthesis." Freely available and open source, Yosys finds use across hobbyist and commercial applications as well as academic.

Note

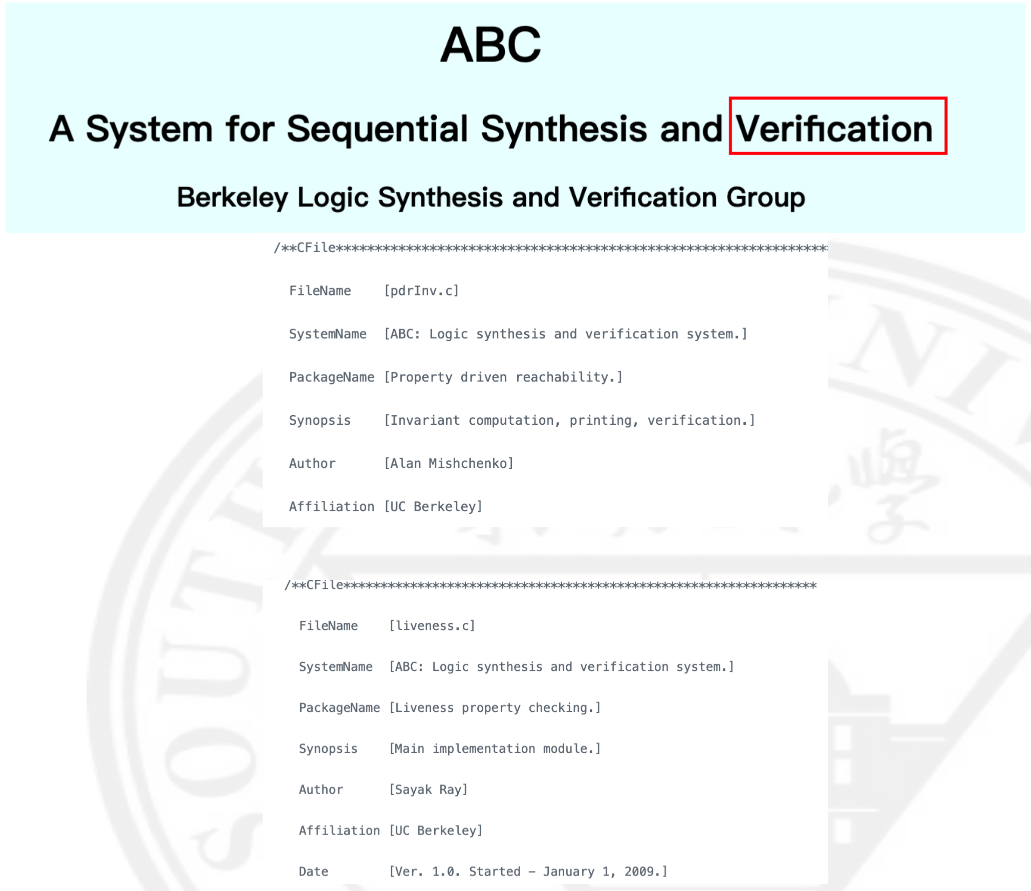
Yosys is released under the ISC License:

A permissive license lets people do anything with your code with proper attribution and without warranty. The ISC license is functionally equivalent to the BSD 2-Clause and MIT licenses, removing some language that is no longer necessary.

Together with the place and route tool nextpnr, Yosys can be used to program some FPGAs with a fully end-to-end open source flow (Lattice iCE40 and ECP5). It also does the synthesis portion for the OpenLane flow, targeting the SkyWater 130nm open source PDK for fully open source ASIC design. Yosys can also do formal verification with backends for solver formats like SMT2.

Yosys, and the accompanying Open Source EDA ecosystem, is currently maintained by Yosys Headquarters, with many of the core developers employed by YosysHQ GmbH. A commercial extension, Tabby CAD Suite, includes the Verific frontend for industry-grade SystemVerilog and VHDL support, formal verification with SVA, and formal apps.

YosysHQ



ABC

A System for Sequential Synthesis and Verification

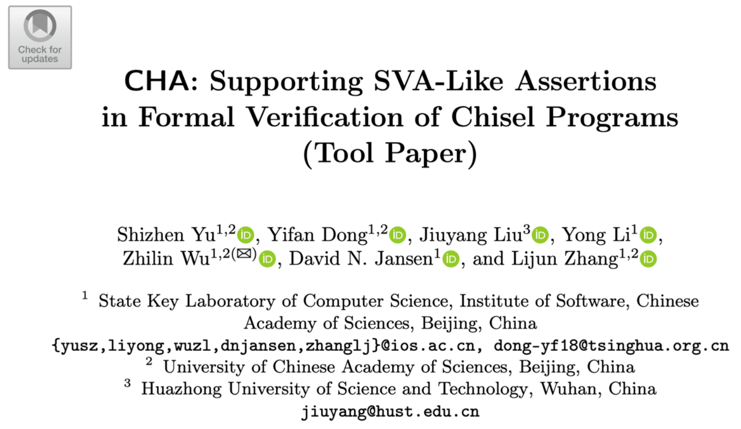
Berkeley Logic Synthesis and Verification Group

```
/**CFile****************************************************************

  FileName    [pdrInv.c]

  SystemName  [ABC: Logic synthesis and verification system.]

  PackageName [Property driven reachability.]

  Synopsis    [Invariant computation, printing, verification.]

  Author      [Alan Mishchenko]

  Affiliation [UC Berkeley]
```

```
/**CFile****************************************************************

  FileName    [liveness.c]

  SystemName  [ABC: Logic synthesis and verification system.]

  PackageName [Liveness property checking.]

  Synopsis    [Main implementation module.]

  Author      [Sayak Ray]

  Affiliation [UC Berkeley]

  Date        [Ver. 1.0. Started - January 1, 2009.]
```

硬件形式化验证工具简介

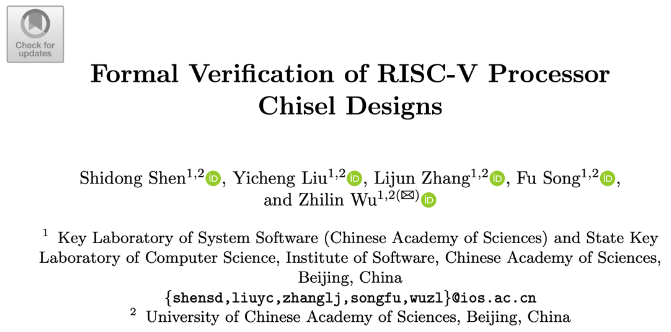
软件所开发相关工具



CHA: Supporting SVA-Like Assertions in Formal Verification of Chisel Programs (Tool Paper)

Shizhen Yu[1,2], Yifan Dong[1,2], Jiuyang Liu[3], Yong Li[1], Zhilin Wu[1,2](✉), David N. Jansen[1], and Lijun Zhang[1,2]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{yusz,liyong,wuzl,dnjansen,zhanglj}@ios.ac.cn, dong-yf18@tsinghua.org.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Huazhong University of Science and Technology, Wuhan, China
jiuyang@hust.edu.cn

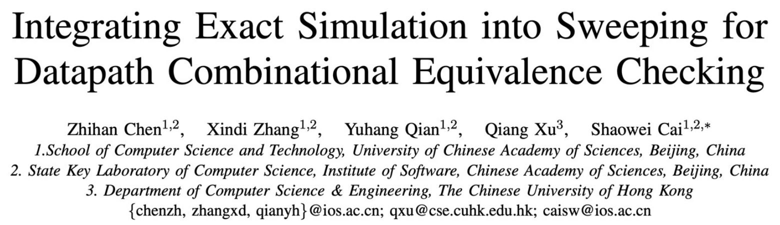


Formal Verification of RISC-V Processor Chisel Designs

Shidong Shen[1,2], Yicheng Liu[1,2], Lijun Zhang[1,2], Fu Song[1,2], and Zhilin Wu[1,2](✉)

[1] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{shensd,liuyc,zhanglj,songfu,wuzl}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

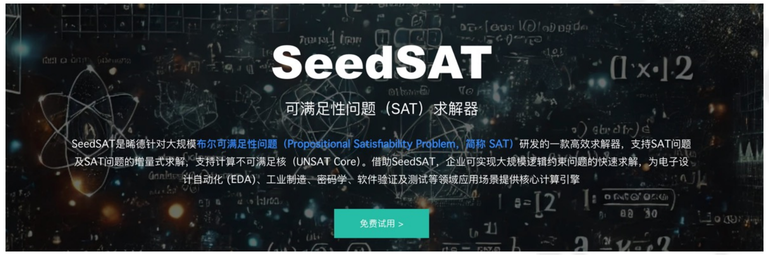
Formal Verification of Chisel



Integrating Exact Simulation into Sweeping for Datapath Combinational Equivalence Checking

Zhihan Chen[1,2], Xindi Zhang[1,2], Yuhang Qian[1,2], Qiang Xu[3], Shaowei Cai[1,2,*]

1.School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
2. State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
3. Department of Computer Science & Engineering, The Chinese University of Hong Kong
{chenzh, zhangxd, qianyh}@ios.ac.cn; qxu@cse.cuhk.edu.hk; caisw@ios.ac.cn



SeedSAT

可满足性问题（SAT）求解器

SeedSAT是晞德针对大规模布尔可满足性问题（Propositional Satisfiability Problem，简称 SAT）研发的一款高效求解器，支持SAT问题及SAT问题的增量式求解，支持计算不可满足核（UNSAT Core）。借助SeedSAT，企业可实现大规模逻辑约束问题的快速求解，为电子设计自动化（EDA）、工业制造、密码学、软件验证及测试等领域应用场景提供核心计算引擎

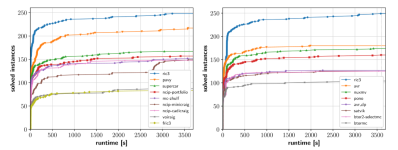
SAT in EDA

rIC3 Hardware Model Checker

License GPLv3 | CI passing | crates.io v1.4.1 | Docker v1.4.1

HWMCC

rIC3 achieved first place in both the bit-level track and the word-level bit-vector track at the 2024 Hardware Model Checking Competition (HWMCC'24).

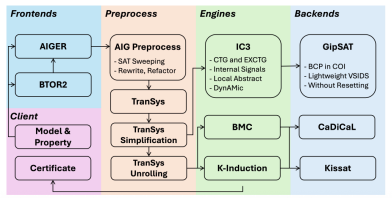


To view the submission for HWMCC'24, please checkout the `HWMCC24` branch or download the binary release at https://github.com/gipsyh/rIC3-HWMCC24.

Publications

- [CAV2025] The rIC3 Hardware Model Checker
- [CAV2025] Deeply Optimizing the SAT Solver for the IC3 Algorithm
- [DAC2024] Predicting Lemmas in Generalization of IC3
- [arXiv] Extended CTG Generalization and Dynamic Adjustment of Generalization Strategies in IC3

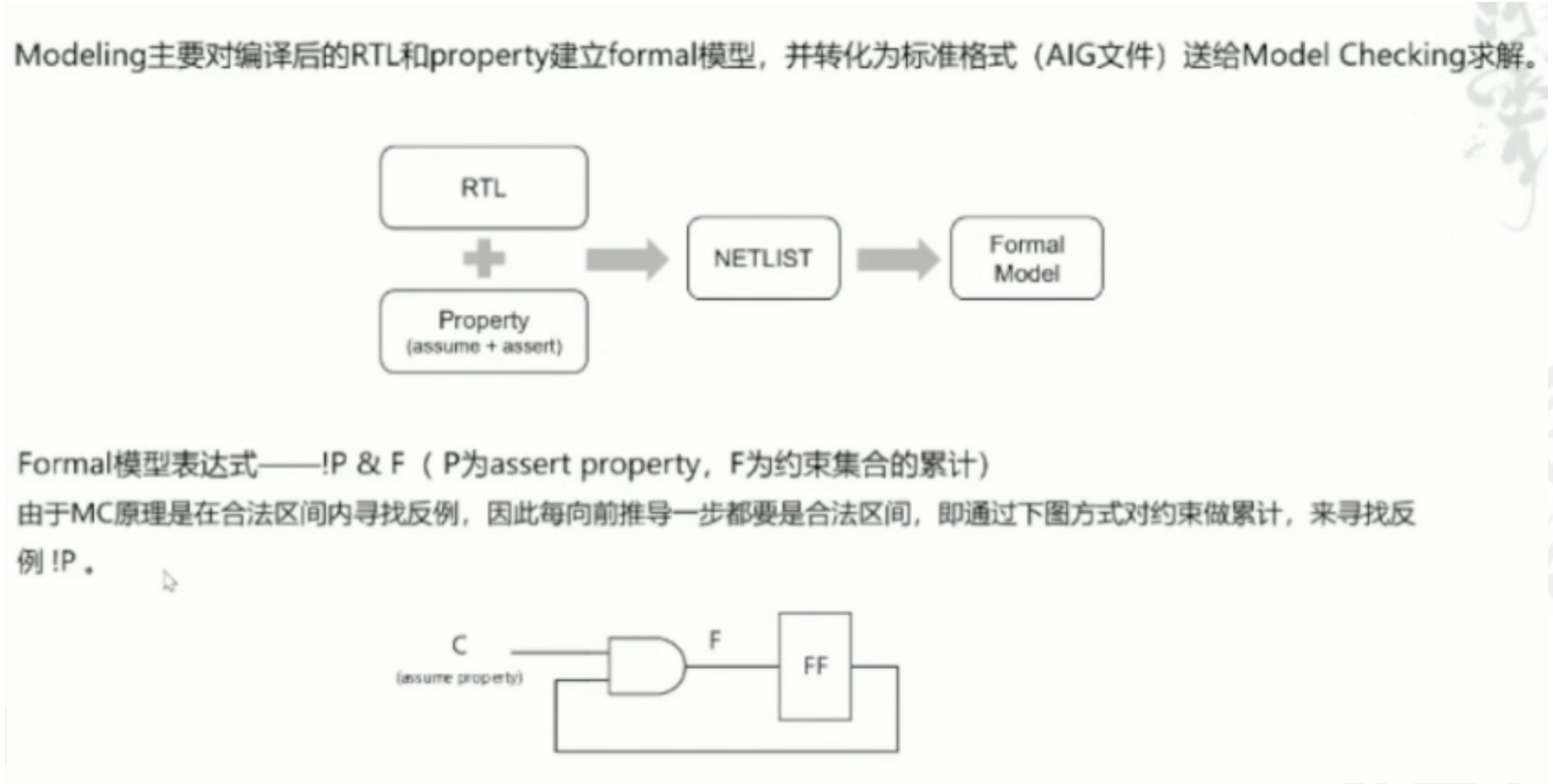
rIC3 Tool Flow



MC in EDA

建模

FPV


Modeling主要对编译后的RTL和property建立formal模型，并转化为标准格式（AIG文件）送给Model Checking求解。

Formal模型表达式——!P & F（P为assert property，F为约束集合的累计）

由于MC原理是在合法区间内寻找反例，因此每向前推导一步都要是合法区间，即通过下图方式对约束做累计，来寻找反例 !P。

src: 2021 CCF-中国软件大会 海思

```
property write_skip;
    @(posedge clk) disable iff (rst)
    !wen |=> $changed(waddr);
endproperty
w_underfill: cover property (write_skip);
```

SVA建模

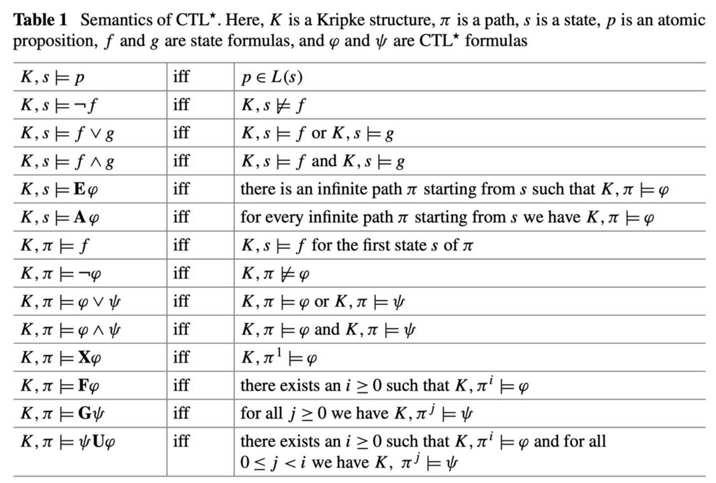


Table 1 Semantics of CTL*. Here, K is a Kripke structure, π is a path, s is a state, p is an atomic proposition, f and g are state formulas, and φ and ψ are CTL* formulas

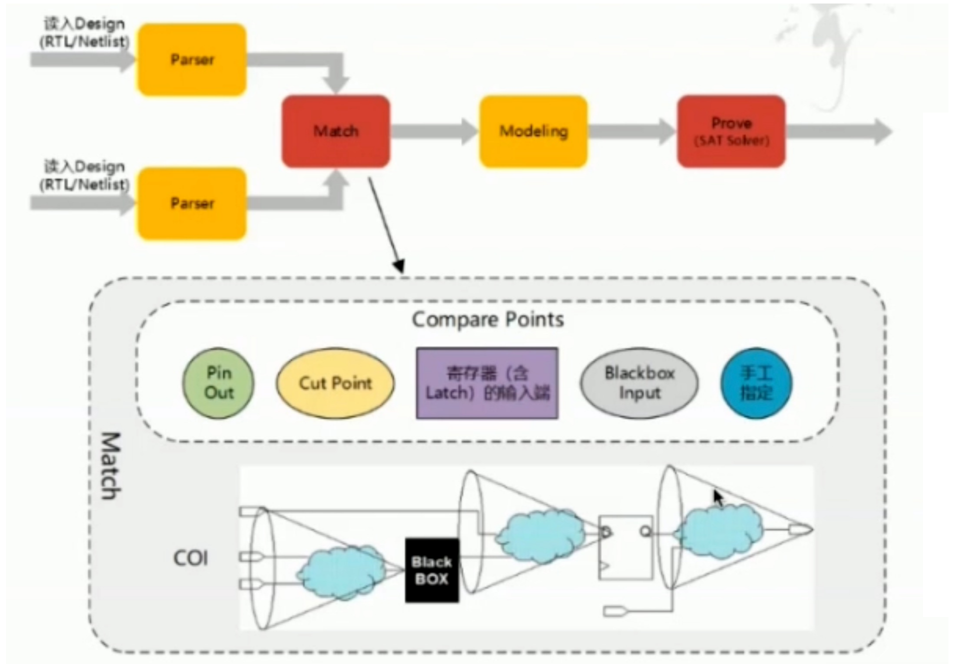
$K, s \models p$	iff	$p \in L(s)$
$K, s \models \neg f$	iff	$K, s \not\models f$
$K, s \models f \vee g$	iff	$K, s \models f$ or $K, s \models g$
$K, s \models f \wedge g$	iff	$K, s \models f$ and $K, s \models g$
$K, s \models \mathbf{E}\varphi$	iff	there is an infinite path π starting from s such that $K, \pi \models \varphi$
$K, s \models \mathbf{A}\varphi$	iff	for every infinite path π starting from s we have $K, \pi \models \varphi$
$K, \pi \models f$	iff	$K, s \models f$ for the first state s of π
$K, \pi \models \neg\varphi$	iff	$K, \pi \not\models \varphi$
$K, \pi \models \varphi \vee \psi$	iff	$K, \pi \models \varphi$ or $K, \pi \models \psi$
$K, \pi \models \varphi \wedge \psi$	iff	$K, \pi \models \varphi$ and $K, \pi \models \psi$
$K, \pi \models \mathbf{X}\varphi$	iff	$K, \pi^1 \models \varphi$
$K, \pi \models \mathbf{F}\varphi$	iff	there exists an $i \geq 0$ such that $K, \pi^i \models \varphi$
$K, \pi \models \mathbf{G}\psi$	iff	for all $j \geq 0$ we have $K, \pi^j \models \psi$
$K, \pi \models \psi\mathbf{U}\varphi$	iff	there exists an $i \geq 0$ such that $K, \pi^i \models \varphi$ and for all $0 \leq j < i$ we have $K, \pi^j \models \psi$

Clarke, Edmund M., et al., eds. *Handbook of model checking*. Vol. 10. Cham: Springer, 2018.

建模

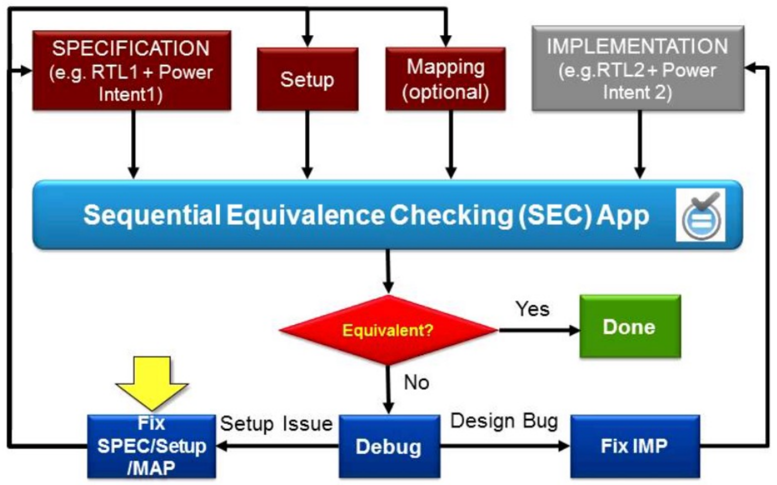
東南大學 SCHOOL OF INTEGRATED CIRCUITS, SEU 集成电路学院

LEC/CEC,C2RTL, etc.



LEC/CEC

src: 2021 CCF-中国软件大会 海思



SPECIFICATION (e.g. RTL1 + Power Intent1)
Setup
Mapping (optional)
IMPLEMENTATION (e.g.RTL2+ Power Intent 2)
Sequential Equivalence Checking (SEC) App
Equivalent?
Yes
Done
No
Fix SPEC/Setup /MAP
Setup Issue
Debug
Design Bug
Fix IMP

SEC

src: Cadence JasperGold

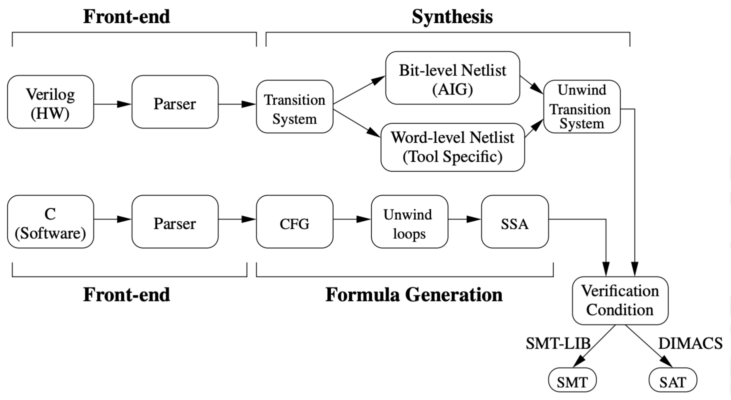
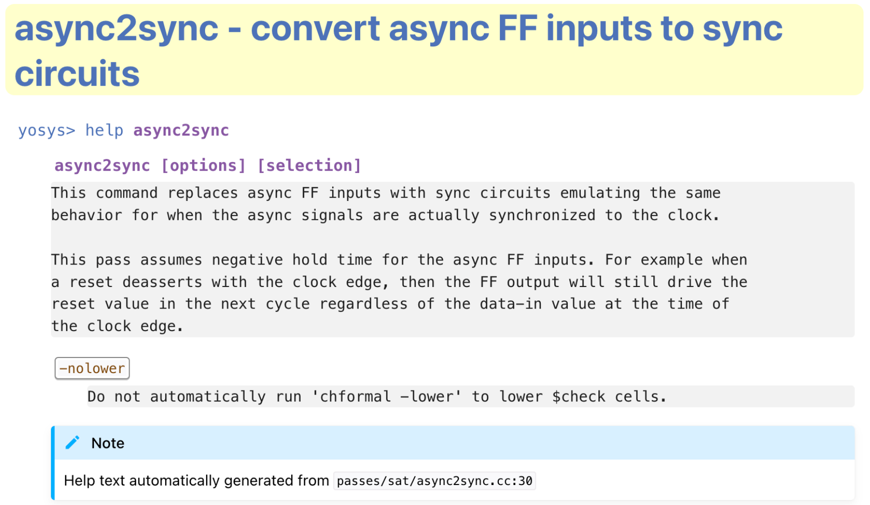


Figure 2: HW-CBMC Tool Flow

Mukherjee, Rajdeep, et al. "Formal techniques for effective co-verification of hardware/software co-designs." *Proceedings of the 54th Annual Design Automation Conference 2017*. 2017.

建模

时钟/复位



async2sync - convert async FF inputs to sync circuits

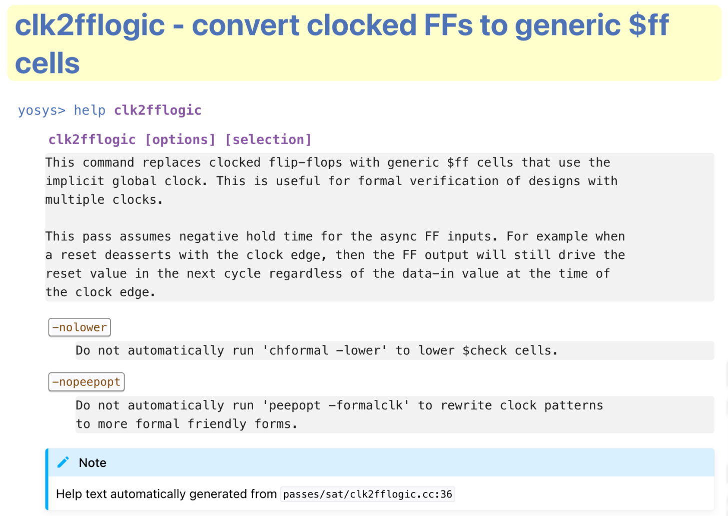
```
yosys> help async2sync

    async2sync [options] [selection]

This command replaces async FF inputs with sync circuits emulating the same
behavior for when the async signals are actually synchronized to the clock.

This pass assumes negative hold time for the async FF inputs. For example when
a reset deasserts with the clock edge, then the FF output will still drive the
reset value in the next cycle regardless of the data-in value at the time of
the clock edge.
```

`-nolower`

```
    Do not automatically run 'chformal -lower' to lower $check cells.
```

Note

Help text automatically generated from `passes/sat/async2sync.cc:30`



clk2fflogic - convert clocked FFs to generic $ff cells

```
yosys> help clk2fflogic

    clk2fflogic [options] [selection]

This command replaces clocked flip-flops with generic $ff cells that use the
implicit global clock. This is useful for formal verification of designs with
multiple clocks.

This pass assumes negative hold time for the async FF inputs. For example when
a reset deasserts with the clock edge, then the FF output will still drive the
reset value in the next cycle regardless of the data-in value at the time of
the clock edge.
```

`-nolower`

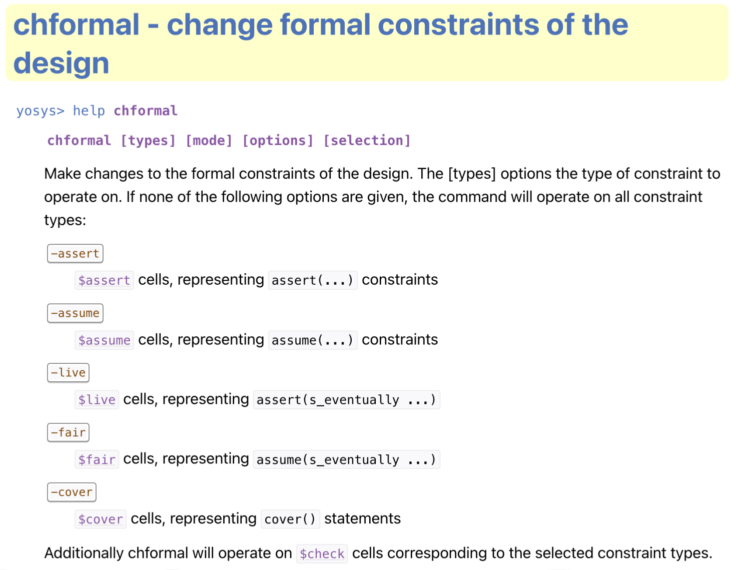
```
    Do not automatically run 'chformal -lower' to lower $check cells.
```

`-nopeepopt`

```
    Do not automatically run 'peepopt -formalclk' to rewrite clock patterns
    to more formal friendly forms.
```

Note

Help text automatically generated from `passes/sat/clk2fflogic.cc:36`



chformal - change formal constraints of the design

```
yosys> help chformal

    chformal [types] [mode] [options] [selection]
```

Make changes to the formal constraints of the design. The [types] options the type of constraint to operate on. If none of the following options are given, the command will operate on all constraint types:

`-assert`
: `$assert` cells, representing `assert(...)` constraints

`-assume`
: `$assume` cells, representing `assume(...)` constraints

`-live`
: `$live` cells, representing `assert(s_eventually ...)`

`-fair`
: `$fair` cells, representing `assume(s_eventually ...)`

`-cover`
: `$cover` cells, representing `cover()` statements

Additionally chformal will operate on `$check` cells corresponding to the selected constraint types.

https://yosyshq.readthedocs.io/projects/yosys/en/latest/cmd_ref.html

综合

RTL层面综合

The Verilog and AST frontends

This chapter provides an overview of the implementation of the Yosys Verilog and AST frontends. The Verilog frontend reads Verilog-2005 code and creates an abstract syntax tree (AST) representation of the input. This AST representation is then passed to the AST frontend that converts it to RTLIL data, as illustrated in Fig. 56.

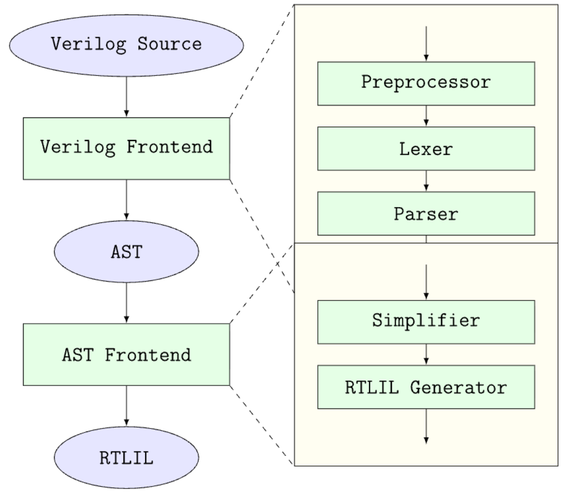
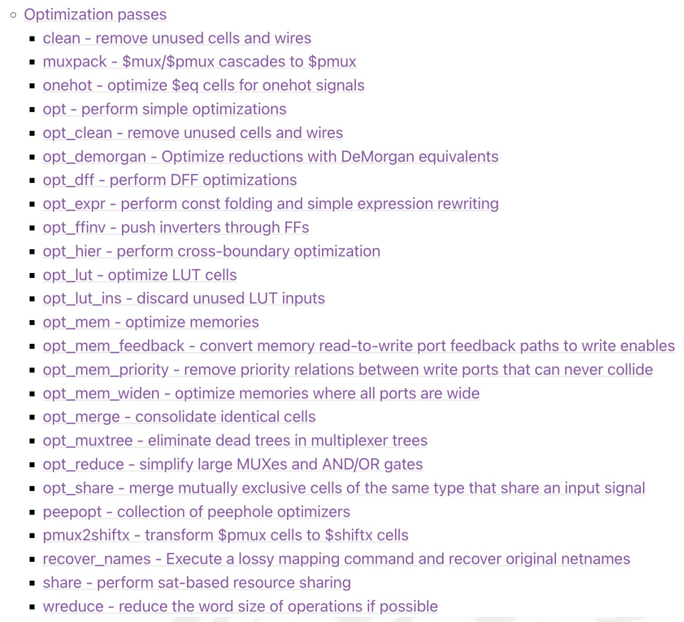
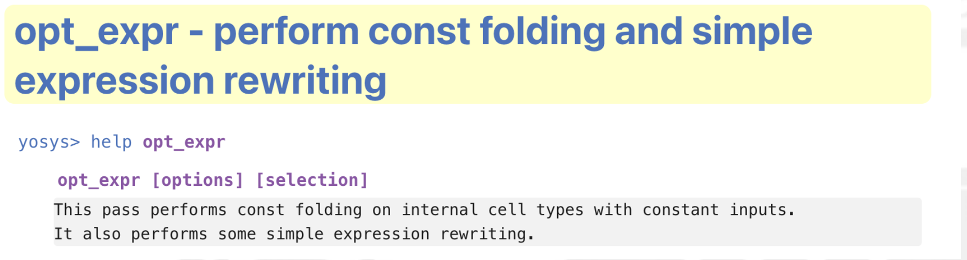


Fig. 56 Simplified Verilog to RTLIL data flow ¶



- Optimization passes
  - clean - remove unused cells and wires
  - muxpack - $mux/$pmux cascades to $pmux
  - onehot - optimize $eq cells for onehot signals
  - opt - perform simple optimizations
  - opt_clean - remove unused cells and wires
  - opt_demorgan - Optimize reductions with DeMorgan equivalents
  - opt_dff - perform DFF optimizations
  - opt_expr - perform const folding and simple expression rewriting
  - opt_ffinv - push inverters through FFs
  - opt_hier - perform cross-boundary optimization
  - opt_lut - optimize LUT cells
  - opt_lut_ins - discard unused LUT inputs
  - opt_mem - optimize memories
  - opt_mem_feedback - convert memory read-to-write port feedback paths to write enables
  - opt_mem_priority - remove priority relations between write ports that can never collide
  - opt_mem_widen - optimize memories where all ports are wide
  - opt_merge - consolidate identical cells
  - opt_muxtree - eliminate dead trees in multiplexer trees
  - opt_reduce - simplify large MUXes and AND/OR gates
  - opt_share - merge mutually exclusive cells of the same type that share an input signal
  - peepopt - collection of peephole optimizers
  - pmux2shiftx - transform $pmux cells to $shiftx cells
  - recover_names - Execute a lossy mapping command and recover original netnames
  - share - perform sat-based resource sharing
  - wreduce - reduce the word size of operations if possible



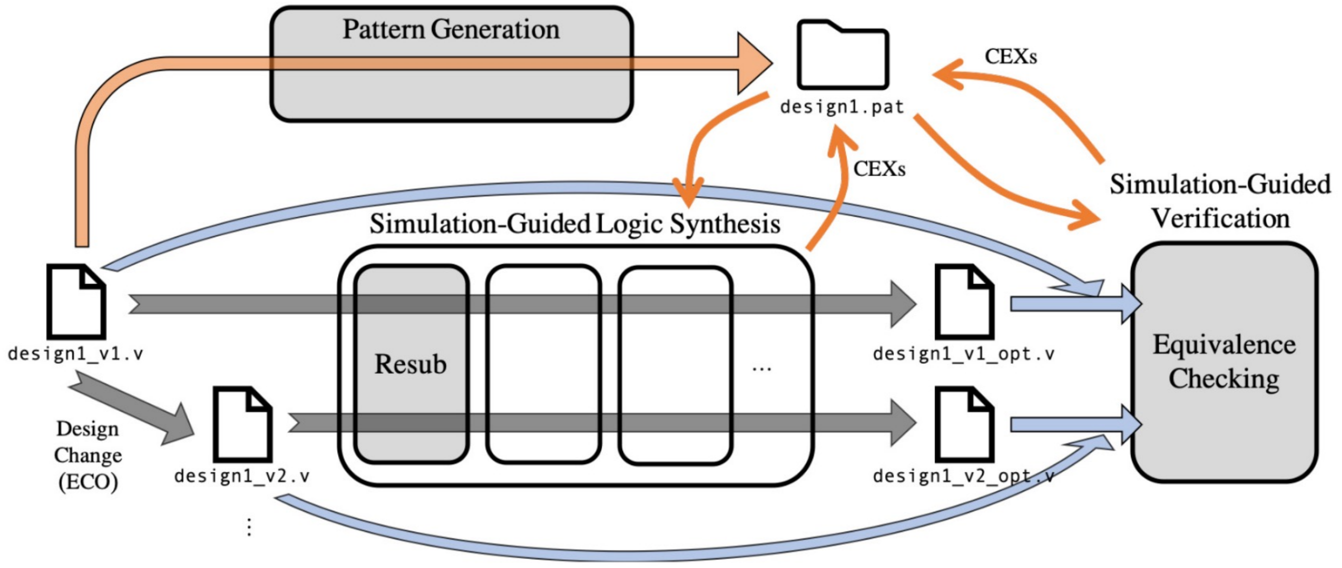
opt_expr - perform const folding and simple expression rewriting

```
yosys> help opt_expr

    opt_expr [options] [selection]

    This pass performs const folding on internal cell types with constant inputs.
    It also performs some simple expression rewriting.
```

https://yosyshq.readthedocs.io/projects/yosys/en/latest/cmd_ref.html

综合

Logic层面综合



Lee, Siang-Yun, et al. "A simulation-guided paradigm for logic synthesis and verification." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.8 (2021): 2573-2586.

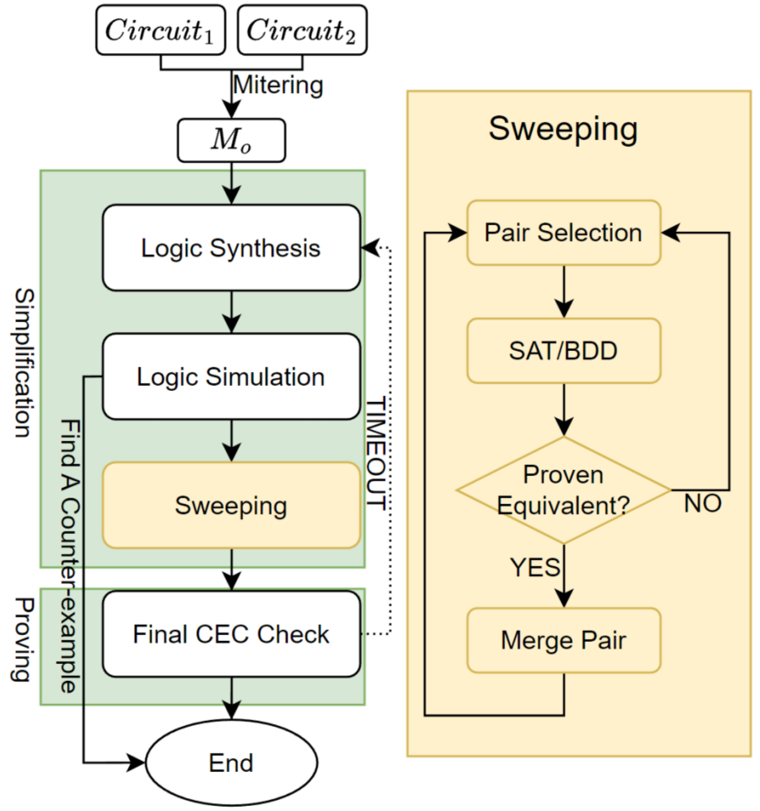
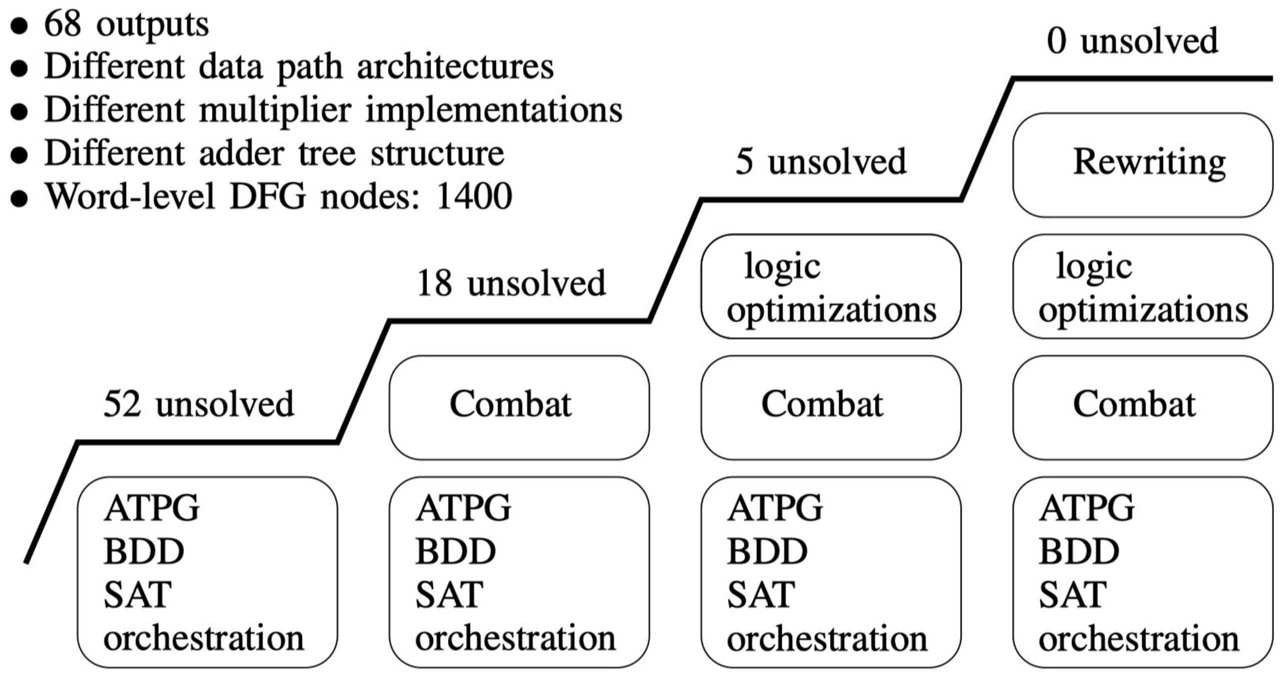


Fig. 2. Framework for a Typical Sweeping Based CEC Algorithm.

Chen, Zhihan, et al. "Integrating exact simulation into sweeping for datapath combinational equivalence checking." *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023.

综合

Logic层面综合



• 68 outputs
• Different data path architectures
• Different multiplier implementations
• Different adder tree structure
• Word-level DFG nodes: 1400
52 unsolved
18 unsolved
5 unsolved
0 unsolved
ATPG BDD SAT orchestration
Combat
logic optimizations
Rewriting

Fig. 4. Effectiveness comes from many techniques

Koelbl, Alfred, et al. "Solver technology for system-level to RTL equivalence checking." *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009.

求解

$p \underset{1}{=} 1$

$p \underset{2}{=} m_1 \oplus m_2$

$m_1 \underset{3}{=} c\ ?\ a_1 : x_1$

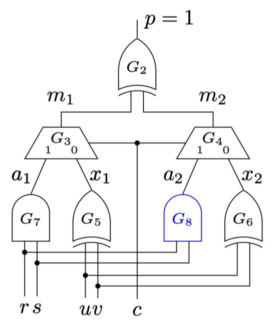
$m_2 \underset{4}{=} c\ ?\ a_2 : x_2$

$x_1 \underset{5}{=} u \oplus v$

$x_2 \underset{6}{=} u \oplus v$

$a_1 \underset{7}{=} r \wedge s$

$a_2 \underset{8}{=} r \wedge s$



$(p)_1$

$(\bar p\, m_1 m_2)_2\ (\bar p\, \bar m_1 \bar m_2)_3\ (p\, \bar m_1 m_2)_4\ (p\, m_1 \bar m_2)_5$

$(\bar m_1 \bar c\, a_1)_6\ (\bar m_1 c\, x_1)_7\ (m_1 \bar c\, \bar a_1)_8\ (m_1 c\, \bar x_1)_9$

$(\bar m_2 \bar c\, a_2)_{10}\ (\bar m_2 c\, x_2)_{11}\ (m_2 \bar c\, \bar a_2)_{12}\ (m_2 c\, \bar x_2)_{13}$

$(\bar x_1 \bar u\, \bar v)_{14}\ (\bar x_1 u\, v)_{15}\ (x_1 \bar u\, v)_{16}\ (x_1 u\, \bar v)_{17}$

$(\bar x_2 \bar u\, \bar v)_{18}\ (\bar x_2 u\, v)_{19}\ (x_2 \bar u\, v)_{20}\ (x_2 u\, \bar v)_{21}$

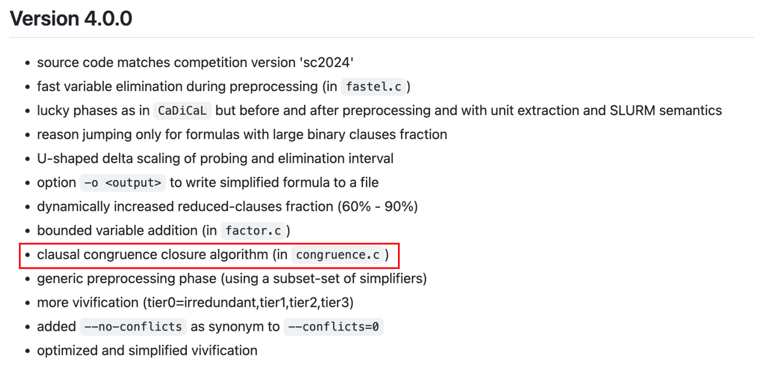
$(\bar a_1 r)_{22}\ (\bar a_1 s)_{23}\ (a_1 \bar r\, \bar s)_{24}$

$(\bar a_2 r)_{25}\ (\bar a_2 s)_{26}\ (a_2 \bar r\, \bar s)_{27}$

(a) gates $G_1, \ldots, G_8$ (b) miter circuit (c) CNF with clauses $C_1, \ldots, C_{27}$

Figure 1 Example of an equivalence checking problem for two identical (isomorphic) circuits consisting each of one AND, XOR, and ITE (multiplexer/if-then-else) gate. The miter circuit in the middle (b) compares the output of the two circuits and assumes they are different by feeding them into another XOR gate which in turn is assumed to produce the output value 1. The equational semantics (a) is shown on the left which after Tseitin encoding [67] gives the CNF (c), e.g., the last AND gate G_8 in the second circuit is encoded by the last three clauses C_{25}, C_{26} and C_{27}.

1）原始电路信息

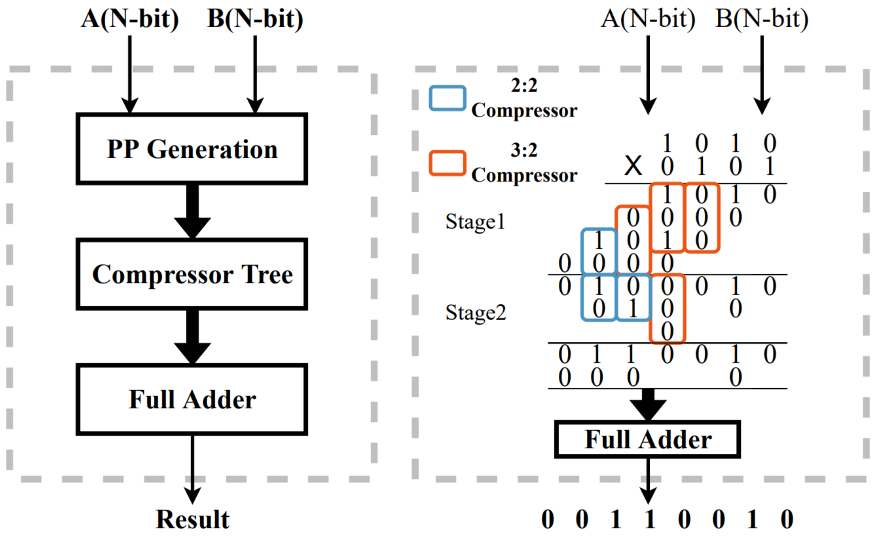


Version 4.0.0

- source code matches competition version 'sc2024'
- fast variable elimination during preprocessing (in `fastel.c`)
- lucky phases as in `CaDiCaL` but before and after preprocessing and with unit extraction and SLURM semantics
- reason jumping only for formulas with large binary clauses fraction
- U-shaped delta scaling of probing and elimination interval
- option `-o <output>` to write simplified formula to a file
- dynamically increased reduced-clauses fraction (60% - 90%)
- bounded variable addition (in `factor.c`)
- clausal congruence closure algorithm (in `congruence.c`)
- generic preprocessing phase (using a subset-set of simplifiers)
- more vivification (tier0=irredundant,tier1,tier2,tier3)
- added `--no-conflicts` as synonym to `--conflicts=0`
- optimized and simplified vivification

https://github.com/arminbiere/kissat/blob/master/NEWS.md

Biere, Armin, et al. "Clausal congruence closure." *27th International Conference on Theory and Applications of Satisfiability Testing (SAT 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.



2）Datapath验证： Symbolic Computer Algebra (RevSCA/AMulet); Theorem Prover and Rewrite(Acl2 + Vescmul)

调度

- Fact1: many solvers, with many parameters to set in each solver
- Fact2: many properties to prove
- Fact3: limited resources, cannot run all tasks at the same time
- Fact4: properties with affinity

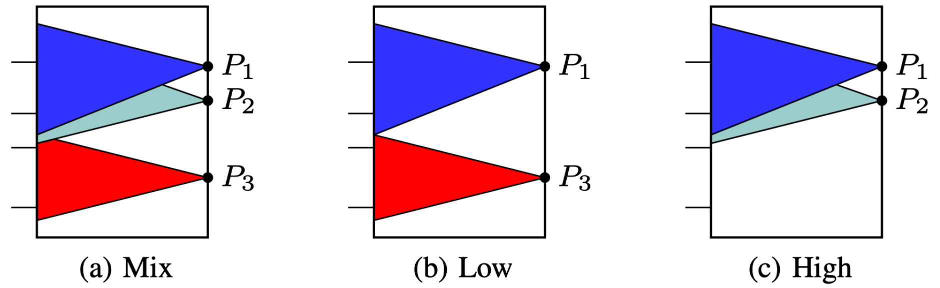
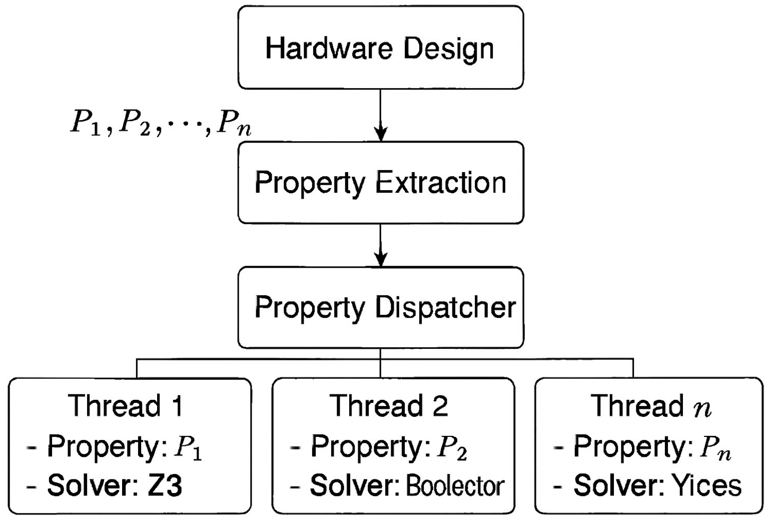


Fig. 1. Cone-of-influence of high- and low- affinity properties.

Dureja, Rohit, et al. "Boosting verification scalability via structural grouping and semantic partitioning of properties." *2019 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2019.



关键问题：

Grouping/Ordering/Assume Guarantee

=>相同时间，相同资源，解出最多的任务

AI时代下硬件形式化工具展望

Natural Language Description

Specification

(Machine Learning Model)

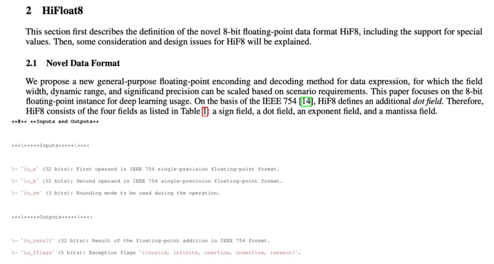
(C Reference Model)

(Chisel)

RTL Code

Netlist

数字后端



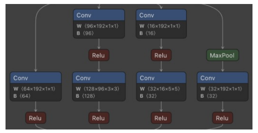
2 HiFloat8

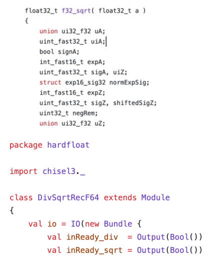
This section first describes the definition of the novel 8-bit floating-point data format HiF8, including the support for special values. Then, some consideration and design issues for HiF8 will be explained.

2.1 Novel Data Format

We propose a new general-purpose floating-point enconding and decoding method for data expression, for which the field width, dynamic range, and significand 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 1: a sign field, a dot field, an exponent field, and a mantissa field.

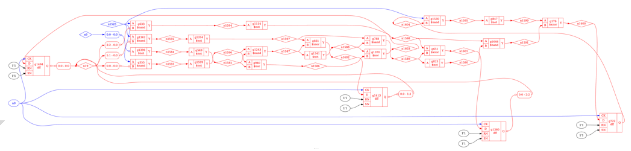
```
**#** **Inputs and Outputs**

***\*****Inputs*****\***:

\- `io_a` (32 bits): First operand in IEEE 754 single-precision floating-point format.
\- `io_b` (32 bits): Second operand in IEEE 754 single-precision floating-point format.
\- `io_rm` (3 bits): Rounding mode to be used during the operation.

***\*****Outputs*****\***:

\- `io_result` (32 bits): Result of the floating-point addition in IEEE 754 format.
\- `io_fflags` (5 bits): Exception flags `{invalid, infinite, overflow, underflow, inexact}`.
```



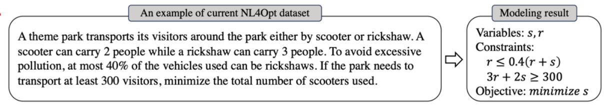


```c
float32_t f32_sqrt( float32_t a )
{
    union ui32_f32 uA;
    uint_fast32_t uiA;
    bool signA;
    int_fast16_t expA;
    uint_fast32_t sigA, uiZ;
    struct exp16_sig32 normExpSig;
    int_fast16_t expZ;
    uint_fast32_t sigZ, shiftedSigZ;
    uint32_t negRem;
    union ui32_f32 uZ;
```

```scala
package hardfloat

import chisel3._

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



基于llm的**智能建模**:
从自然语言描述中生成形式化Spec



An example of current NL4Opt dataset

A theme park transports its visitors around the park either by scooter or rickshaw. A scooter can carry 2 people while a rickshaw can carry 3 people. To avoid excessive pollution, at most 40% of the vehicles used can be rickshaws. If the park needs to transport at least 300 visitors, minimize the total number of scooters used.

⇨

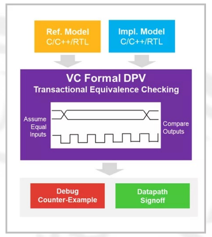
Modeling result

Variables: s, r
Constraints:
$r \leq 0.4(r+s)$
$3r + 2s \geq 300$
Objective: *minimize* s

更多可能

基于llm的Coding Agent:
从规格文档直接生成正确的C代码/Chisel代码/RTL代码

基于llm+graph的Agent:
从规格文档直接生成正确的网表



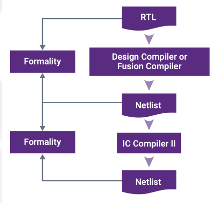


Figure 1: Formality equivalence checking solution

AI可以改变当前瀑布式设计流程
=>设计左移

LLM时代下新的设计范式定义了
新的验证问题！

东南大學 SCHOOL OF INTEGRATED CIRCUITS, SEU 集成电路学院

LLM时代下硬件形式化验证展望

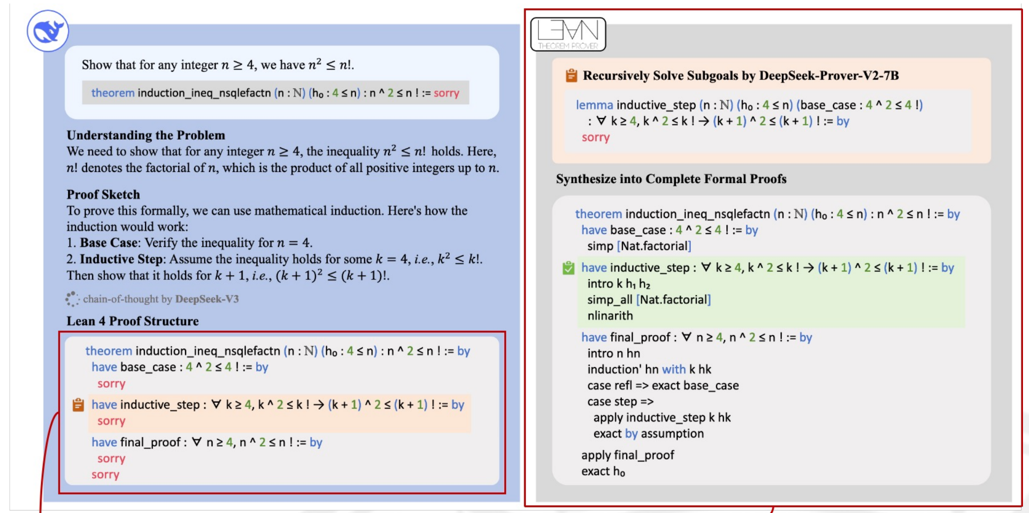
Time Cost for C2RTL verification (setup to fully proven)

- TCL refinement loop to run correctly
    - fix constraints to run correctly
    - fix C&RTL input/output mapping
- TCL refinement loop to converagence
    - try case spliting, 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:
浮点除法用例：2000+ tcl验证环境，花一年才达到完备收敛

src: https://www.bilibili.com/video/BV1fJ23YQE9Y/?spm_id_from=333.1387.favlist.content.click&vd_source=637e596cabafb504f6a386a96f7ec263



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

谢谢聆听

- 感谢2021 CCF-中国软件大会/全国形式化方法与应用会议相关议题提供思路；
- 感谢华为海思相关项目组提供宝贵落地经验；
- 感谢胡旭东同学对材料的贡献；

拥抱开源，合作共赢！

课题组主页：https://formind.netlify.app/