第四届（2022）集成电路 EDA 设计精英挑战赛

赛题指南

一、 赛题名称：高位宽运算电路的逻辑等价性验证

二、 命题企业：华为海思半导体/诺亚方舟实验室

三、 赛题 Chair：储著飞 宁波大学、郭静静 南京邮电大学

四、 赛题背景

数字电路中的逻辑等价性验证（Logic Equivalence Checking, LEC），是一种典型的形式化验证方法。其主要作用是验证两个 design 是否等价，可以在两个 RTL 之间或者两个 netlist 之间，也可以在 RTL 和 netlist 之间，如图 1 所示。分布式并行是 EDA 解决速度瓶颈的一个重要方向，同时也是 EDA 工具部署云端平台的关键技术。分布式计算把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给多个计算机进行处理，最后把这些计算结果综合起来得到最终的结果。这样就可以节约整体计算时间，大大提高计算效率。

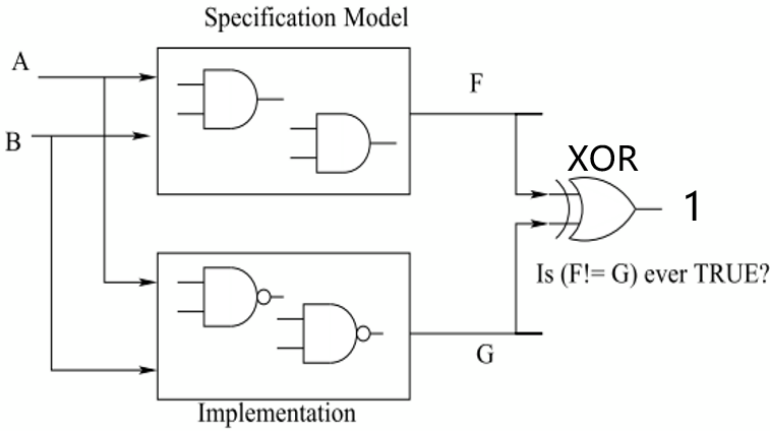

图 1. LEC 的标准模型。

LEC 在整个 EDA 流程中，是高频调用的模块。在当前的学术研究中，最主流的方法是将待验证 miter（将待验证的两个 design 通过 XOR 连接构成的组合电路）转成 SAT（布尔可满足性）表达式（如参考文献 1-2，及详解部分），而后直接调用 SAT 求解器求解。除此以外，也有通过 BDD（binary decision diagram，二元决策图）、ATPG（automatic test pattern generation，自动测试向量生成）等方法来完成求解的学术研究（如参考文献 3-6）。

目前国际上应用最广的 LEC 工具包括 Synopsys 的 Formality 和 Cadence 的 Conformal。国内则有奥卡思、EasyLogic 等公司有对应的 LEC 工具。在 LEC 验证中，高位宽复杂运算电路尤其是高位宽乘法器的验证一直是非常困难的问题之一。这些电路往往具备 XOR 级联、PI 占比高、连接关系复杂等特点，导致计算超时严重。

从各种形式验证手段背后思想、算法的角度来看，不管功能等价性验证（model checking）、定理证明（theorem proving）还是 LEC 它们之间很多概念、求解思路和算法框架可以共享，因此对 LEC 中乘法器的验证提速，会极大的促进整个领域对乘法器尤其是复杂运算电路的证明的理解。但是，LEC 的背景相对清晰，因此，参赛者只要有基本的逻辑验证基础，即可完成求解，包括但不限于电子信息工程、计算理论（SAT 求解）、应用数学等。

五、 赛题描述

在本次比赛中，LEC 工具的 input 是 miter 电路的 AIG（只有 AND gate 和 NOT gate，其文件后缀是 aag），输出是 UNSAT（证明两个 design 等价）或者找到的解（证明两个 design 不等价）。AIG 基本模型如图二所示。

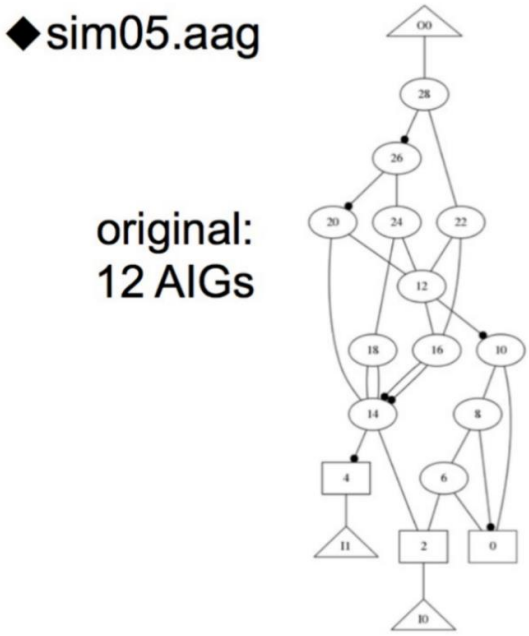

图 2：AIG 的基本样例。

一般来讲，LEC 的求解可以有多种方式：

a) 将 AIG 格式文件转换成 SAT 求解器可以接受的 CNF 格式，从而直接调用 SAT 求解器来求解。这里，CNF 是指一个组合取表达式，例如，图 3 所展示的样例电路，可以转成图 4 的 CNF 格式：

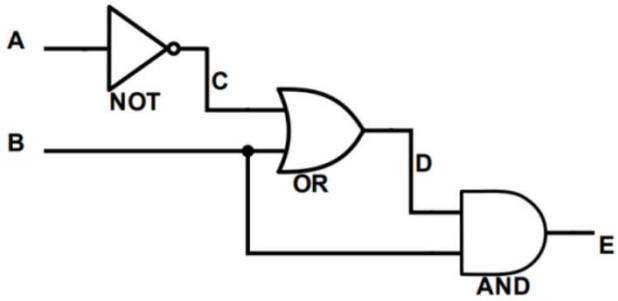

图 3、样例电路

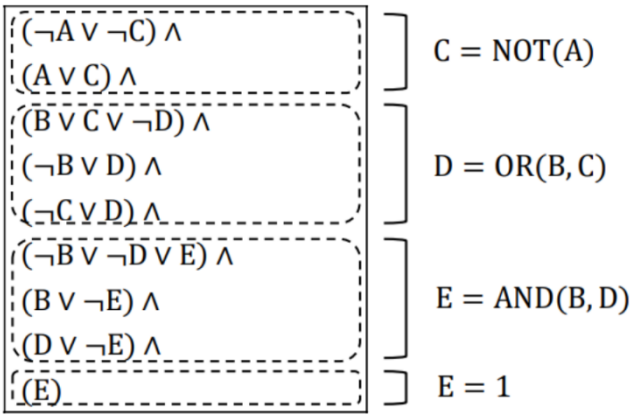

图 4、对应的 CNF 表达式。

这里，AND 和 NOT 与 CNF 的一般对应关系如下：

$$AND(x_1, x_2, \ldots, x_n) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$$

$$NOT(x) = \neg x$$

在读取过程中，CNF 的表达格式一般采取 DIMACS 格式，一般包含 comments, problem line, clauses 几部分，其具体格式如图 5 所示。

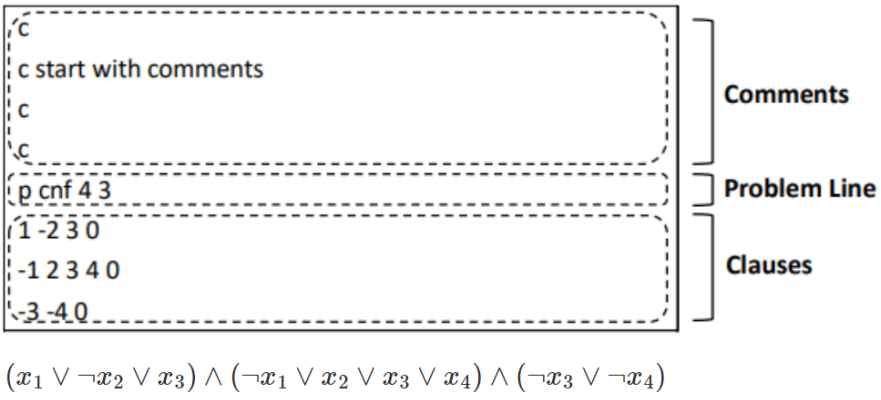

图 5、CNF 表达式的一般存取格式

b）在 AIG 上直接求解，或者转成 BDD 等其他格式。如果求解器是支持 AIG 的表达格式的，也可以直接从 AIG 上完成证明，或者换成所提交的求解器支持的格式，具体格式形式本次比赛并不限制，但是转化时间会计入总的证明时间。

c）本次比赛也支持采用定理证明等高阶方法完成证明，所需要的具体格式或者信息需要参赛团队自行分析 AIG。

在本次比赛中，我们既鼓励参赛队员自己转成需要的格式，也鼓励直接调用开源工具，例如，如果需要借助 SAT 完成求解，鼓励大家直接调用开源工具 ABC（http://people.eecs.berkeley.edu/~alanmi/abc/ ）直接完成转化，以转成 CNF 为例，具体的步骤很简单：read AIG；write_cnf xxx.cnf 即可保存成需要的 CNF 格式。值得一提的是，ABC 支持的转化格式很多，如下图所示。

write – Writes the output file using one of the available file writers. The file extension is used to determine what file writer to invoke. The recognized file extensions are: *aig, baf, bench, blif, cnf, dot, eqn, gml, pla, verilog.*

write_aiger – Writes the combinational AIG in binary AIGER format developed by Armin Biere. This format is very compact and leads to a substantial reduction in the reading/writing times. (When writing AIGER for sequential circuits with non-0 initial states, use command *zero* to normalize the registers initial states.)

write_baf – Writes the combinational AIG in Binary Aig Format (BAF). For a description of BAF, refer to the source code file *src/base/io/ioWriteBaf.c*. This format is superseded by the AIGER format and kept for backward compatibility with earlier versions of ABC.

write_bench – Outputs the current network into a BENCH file.

write_blif – Outputs the current network into a BLIF file. If the current network is mapped using a standard cell library, outputs the current network into a BLIF file, compatible with SIS and other tools. (The same genlib library has to be selected in SIS before reading the generated file.) The current mapper does not map the registers. As a result, the mapped BLIF files generated for sequential circuits contain unmapped latches. Additionally, command *write_blif* with command-line switch *–l* writes out a part of the current network containing a combinational logic without latches.

write_blif_mv – Outputs the current network into a BLIF-MV file. Two write a hierarchical BLIF-MV output, use command *write_hie*.

write_cnf – Outputs the current network into a CNF file, which can be used with a variety of SAT solvers. This command is only applicable to combinational miter circuits (the miter circuit has only one output, which is expected to be zero under all input combinations).

write_counter – Outputs the counter-example after solving a satisfiable miter using commands *sat*, *prove* or *iprove* .

write_dot – Outputs the structure of the current network into a DOT file that can be processed by graph visualization package GraphViz. Currently work only if the current network is an AIG.

write_eqn – Outputs the combinational part of the current network in the Synopsys equation format.

write_hie – Outputs the hierarchy containing black boxes into a file if the original design contained black boxes. The original file should be given as one of the arguments in this command.

write_gml – Outputs the structure of the current network into a GML file used by some graph editors, such as yEd, a free product of yWorks.

write_pla – Outputs the current network into a PLA file. The current network should be collapsed (each PO is represented by a node whose fanins are PIs). Works only for combinational networks.

write_verilog – Outputs the network using technology-independent Verilog.

参赛者可以自行选择需要的。当然也支持利用其他开源软件完成转化，只是需要在提交的文档 readme 中做相应说明（针对格式以及使用方式等）。

当然如果求解器是直接针对 AIG 去完成的，则不需要转化步骤。但无论用什么方式求解，本次比赛都支持在参赛者所支持的形式上做化简，但化简时间算作整体

时间以内。

六、 评分标准

本次比赛需要提交一个 zip 格式的压缩包，其中至少包含：可以直接运行的 run build.sh、编译好的 code 二进制文件以及 readme。其中，readme 是对算法的基本说明，包括允许的输入格式说明等。如果文件脚本运行是需要依赖第三方库的（例如 ABC 等开源软件），需要一并提供，防止由于版本不同而带来结果差异。（可以脚本包装）

比赛结果需要验证正确性，如果是对应 SAT 问题，则需要输出解，具体要求是：按照 AIG 的输入顺序，输出对应的赋值向量（文本表示，字符串，每位对应一个输入的取值）。我们不排除比赛提供电路中，存在某一个位的取值不影响结果（即取值 x，即：0 或者 1 都不影响最后正确性），参赛者可以自行命该赋值 0 或者 1，都不会影响正确性验证。

本赛题将提供 200 个测试用例，并按照简单、中等、困难的比例是 2:5:3 的比例构成（40 个简单问题，100 个中等问题，60 个困难问题）。所有测试用例在比赛结束前不对参赛者公开。为方便所有参赛者验证算法和方案的正确性以及调优结果，本次比赛会提供 8 公开测试用例，这 8 个测试用例的按照简单、中等、困难三个等级分类，但是这 8 个测试用例不参与最后的测试排名。

本次比赛测试系统 Linux CentOS 7，gcc 版本>=7.5，内存不低于 128GB。

本次比赛将按照在评分的具体计算如下：

1、所有的证明方案（即：参赛者所提交的二进制文件）针对单一测试用例的测试都将在 100s 内完成验证，超过 100s 没有完成证明的则被视为超时，超时问题不

被计算排名。同时，本次比赛将限制不同参赛方案在单一测试用例中的最大内存（不可以超过 16GB，否则按照超时统计）

2、相同时间内计算的正确结果多的，评分越高。考虑到测试用例将按照简单、中等、困难三类问题组成，完成不同难度的测试用例的证明难度不同，因此，最终的计算个数将按照如下公式完成统计：计算正确数目 =1* Numeasy+2*Nummid+3*Numhard

3、正确个数相同，就按照计算时间更小的，排名更靠前。同样的额，考虑的不同问题的难度，计算个数相等的参赛方案将按照如下公式完成统计时间：计算时间 =3* Timeeasy+2*Timemid+1*Timehard

4、超过 1%的问题计算错误，则退出排名。

七、 声明：

获胜者所提交的二进制文件（包括相关的说明文档等）将在比赛结束后删除，本次比赛不做任何形式的保留。所有版权归获胜者所有。所有参赛者的二进制文件需要符合 MIT license。本次比赛所涉及的所有比赛题目也将在比赛后删除，不做任何形式的保留。所有参赛者提供的二进制文件、文档以及本次比赛的题目仅供本次比赛使用，不做任何形式的商用。如果比赛结束后有任何知识产权争议，将交由对应的委员会进行裁决。

八、 参考资料：

[1] Goldberg, Evguenii I., Mukul R. Prasad, and Robert K. Brayton. "Using SAT for combinational equivalence checking." Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001. IEEE, 2001.

[2] Arora, Rajat, and Michael S. Hsiao. "Enhancing sat-based equivalence checking with static logic implications." Eighth IEEE International High-Level Design Validation and Test Workshop. IEEE, 2003.

[3] Reda, Sherief, Rolf Drechsler, and Alex Orailoglu. "On the relation between SAT and BDDs for equivalence checking." Proceedings International Symposium on Quality Electronic Design. IEEE, 2002.

[4] Lajaunie R P, Hsiao M S. An effective and efficient ATPG-based combinational equivalence checker[C]//Proceedings of the 15th ACM Great Lakes symposium on VLSI. 2005: 248-253.

[5] K.L. McMillan, "Symbolic Model Checking: An Approach to the State Explosion Problem", Kluwer Academic Publishers, 1993.

[6] K-T. Cheng and A. Kristic, "Current Directions in Automatic Test-Pattern Generation", Computer, vol. 32, no.11, IEEE Computer Soc., Nov, 1999.

详解:

SAT 表达式

计算机里的位变量都是布尔变量，只取值 0 或 1。两个布尔变量之间的运算只有“与”“或”“非”，表示为“∧”“∨”“¬”。例如看两个逻辑函数 f1 和 f2 是否等价，只要判断（f1∧¬f2）∨(¬f1∧f2)是否永远不能被满足，我们可以发现 f1 和 f2 的关系就是异或。