PhyLS Software User Guide

AI-driven Physically Aware Synthesis Platform

Doc No. 001-10001 Rev. 2A

目录

修订历史

下表列出了文档版本之间的变更。

修订版本	产品版本	描述
0A	PhyLS 2025.10	初始版本
0B	PhyLS 2026.01	初始版本修订稿
1A	PhyLS 2026.03	更新为命令行参数配置，新增工作模式说明和推荐流程
2A	PhyLS 2026.04	新增 Area 模式、Effort 等级、层次化设计支持；更新参数表和推荐流程

前言

本前言介绍了 PhyLS 软件用户指南。

预期读者

本指南适用于熟悉数字 ASIC 设计流程、逻辑综合、物理实现（布局布线）概念的工程师。用户应熟悉标准的 EDA 输入格式，包括 Verilog、SDC、LEF 和 LIB。

约定

本书中使用的排版约定如下表所述。

样式	描述
<placeholder>	在文件路径、目录和命令中，可替换的项目使用斜体并用尖括号<>括起来。
Code block	命令、文件名、目录名、参数名和代码示例以等宽字体显示。

1 简介

本章介绍 PhyLS 软件并概述其在设计流程中的位置。

1.1 产品概述

PhyLS 是一个 AI 原生、物理感知综合平台，旨在加速设计收敛并交付高质量的 QoR。为了弥合前端逻辑综合与后端物理实现之间的关键差距，PhyLS 在后综合阶段引入物理信息，对映射的门级网表进行重综合。PhyLS 通过生成物理设计友好的网表来消除成本高昂的后期设计迭代。

PhyLS 以一个轻量级、无缝集成的插件形式提供，可融入任何现有的 EDA 工具链，包括商业工具流（如 Synopsys、Cadence）或开源流程。

本版本（2026.04）新增了**面积驱动优化模式**（Area Mode），使工程师可以在维持时序预算的前提下最小化标准单元面积；支持**层次化设计**（--hier）和可配置的**优化力度**（--effort）。

1.2 核心优势

- **卓越的 QoR：** 与业界领先的商业工具相比，可将最终版图的 PPA 降低超过 15%。
- **快速设计收敛：** 利用预测性物理感知技术减少迭代次数，将设计收敛速度提高多达 40%。
- **多目标优化：** 支持时序驱动（Timing）和面积驱动（Area）两种优化模式，覆盖不同设计阶段需求。
- **自适应优化力度：** 提供 High / Mid / Low 三档 Effort，平衡结果质量与运行时间。
- **无缝集成：** 作为轻量级插件，即插即用，与现有工具链兼容。
- **标准格式支持：** 支持包括 Verilog、SDC、LEF 和 LIB 等标准格式。

1.3 PhyLS 在 EDA 流程中的位置

PhyLS 旨在作为 Logic Synthesis 和 Placement 之间的桥梁，扮演一个**物理感知重综合**的角色。

传统流程通常如下：

逻辑综合 → 综合后网表 (.v) → 布局规划 (Floorplan) → 布局 (Placement) → 布线 (Routing)

集成 PhyLS 后的流程：

逻辑综合 → 综合后网表 (.v) → 布局规划 (Floorplan) → **PhyLS (物理感知重综合)** → 重综合网表 (.v) → 布局 (Placement) → 布线 (Routing)

PhyLS 利用 **Floorplan** 提供的物理信息（如 IO、宏单元位置和边界），在执行 Global Placement 之前，对网表进行优化，以生成对物理实现更友好的网表，从而减少时序收敛的迭代次数。

2 工具输入

本章详细描述了运行 PhyLS 所需的所有输入文件及其格式要求。

2.1 输入文件概览

PhyLS 的运行需要以下几类文件：

1. **设计文件 (Design Files)**
   - <design_name>.v：综合后的门级网表。
   - <design_name>.def 或 <design_name>.def.gz：物理信息文件。
   - <design_name>.sdc：综合时序约束文件。
2. **平台文件 (Platform / PDK Files)**
   - *.lib：时序和功耗库文件。
   - *.lef：物理库文件（Tech LEF & Cell LEF）。
   - setRC.tcl：寄生参数定义文件。

2.2 目录结构要求

PhyLS 要求输入文件按特定的目录结构进行组织。

2.2.1 平台 (PDK) 目录

platform 目录是 PDK 所在目录，必须包含 lib 和 lef 文件夹。

目录结构示例（以 ASAP7 为例）：

```
# ls -R /platform/ASAP7/

/platform/ASAP7/:
lef/  lib/  setRC.tcl

/platform/ASAP7/lef:
asap7_tech_1x_201209_tech.lef
asap7sc7p5t_28_L_1x_220121a.lef
asap7sc7p5t_28_R_1x_220121a.lef
...

/platform/ASAP7/lib:
asap7sc7p5t_SIMPLE_LVT_TT_nldm_211120.lib
asap7sc7p5t_SIMPLE_RVT_TT_nldm_211120.lib
sram_asap7_16x256_1rw.lib
...
```

注： lef 目录下必须有且只有一个 *_tech.lef 技术文件。

2.2.2 设计目录

designs 是设计所在目录。每个设计占据一个子目录，目录名即 design_name。该目录必须包含网表、SDC 和 DEF 文件。

目录结构示例（如果 design_name 为 ChipTop）：

```
# ls /designs/ChipTop/

ChipTop.sdc  ChipTop.v  ChipTop.def
```

注：

1. .v、.sdc 和 .def 文件的**基本名称必须**与 design_name 的值完全一致。
2. design_name **必须**与 top module 的值完全一致。
3. DEF 文件可为 gzip 压缩后的 .def.gz 文件。

2.3 详细输入文件说明

2.3.1 门级网表 (Verilog)

- **文件：** <design_name>.v
- **描述：** 这是 PhyLS 的主要操作对象。它是一个**综合后的**、已经映射到标准单元库的门级 Verilog 网表。PhyLS 将对此网表进行重综合。

2.3.2 布局规划文件 (DEF)

- **文件：** <design_name>.def / <design_name>.def.gz
- **描述：** DEF 文件可以是以下两种类型之一：
  - **Floorplan DEF：** 仅包含芯片的物理边界、IO 端口位置、宏单元布局，但不包含标准单元的布局。PhyLS 将基于此信息执行时序驱动的全局布局。
  - **Placed DEF：** 包含所有标准单元的布局位置和朝向。PhyLS 将保留现有布局并进行增量优化。这是一个**布局后**的 DEF 文件。
- **必须包含：**
  - 芯片的物理边界。
  - 所有 IO 端口的固定位置。
  - 所有宏单元和标准单元的布局位置和朝向。
- PhyLS 利用这些物理信息来估算线长、寄生参数和拥塞，从而指导其优化决策。

2.3.3 时序约束 (SDC)

- **文件：** <design_name>.sdc

- **描述：** 标准的 SDC 文件。它定义了设计的时序目标，PhyLS 将依据这些约束来优化时序。
- **关键内容：**
  - 时钟定义 (create_clock)。
  - 输入/输出延迟 (set_input_delay, set_output_delay)。
  - 端口的过渡时间 (set_input_transition) 和负载 (set_load)。
  - （可选）伪路径 (set_false_path) 或多周期路径 (set_multicycle_path)。

2.3.4 库文件 (LIB/LEF)

- **文件：** <platform>/lib/*.lib，<platform>/lef/*.lef
- **限制：** 当前版本仅支持单一工艺角 (Single Process Corner)。
- **描述：** 这是 PDK 的标准库文件。
  - **.lib：** 时序库文件。提供标准单元信息，用于计算 Delay、Slew、Power 和时序检查（Setup/Hold）。
  - **.lef：** 物理库文件。包含工艺 LEF (techlef) 和单元 LEF。

2.3.5 RC 信息 (setRC.tcl)

- **文件：** <platform>/setRC.tcl
- **描述：** 此文件用于设置每个金属层对应的 RC 信息，PhyLS 会使用这些信息来计算寄生参数和时序指标。
- **内容：** setRC.tcl 包含一系列 set_layer_rc Tcl 指令。

如何生成 setRC.tcl：

PhyLS 工具目前无法直接读取 .qrc 等工艺角文件。作为一种解决方法，您可以使用 P&R 工具来生成所需的 RC 值。

步骤 1：从 P&R 工具中报告 RC 值

在 P&R 工具中读入 .qrc 文件后，使用以下 Tcl 指令来报告所有金属层的单位寄生参数：

```
# 设置所有需要报告的金属层
set all_layers { M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 AP }

# 设置您关心的 RC corner
set rc_corner "ssg0p81v125c.setup_rc"

foreach layer $all_layers {
  puts "\n--- Layer: $layer (Corner: $rc_corner) ---"
  report_unit_parasitics -layer $layer -rc_corner $rc_corner
}
```

步骤 2：查看 P&R 工具输出

上述指令将生成类似如下的报告：

```
--- Layer: M1 (Corner: ssg0p81v125c.setup_rc) ---
Unit parasitics for layer M1 (in fF,ohm)
RC-Corner = ssg0p81v125c.setup  rc
  Cap = 0.1
  Res = 1.0
Parasitics for default Via "VIA1"
  Via Cap = 0.01
  Via Res = 1.00
```

步骤 3：转换为 setRC.tcl 格式

您需要将 P&R 工具报告的值转换为 PhyLS setRC.tcl 文件所需的格式和单位。set_layer_rc 指令的单位要求如下：

- -capacitance：单位为 **pF**
- -resistance：单位为 **kOhm**

根据上一步的输出 (Cap = 0.1 fF, Res = 1.0 ohm)，转换如下：

- Capacitance: 0.1 fF = 0.0001 pF
- Resistance: 1.0 ohm = 0.001 kOhm

对应的 setRC.tcl 指令即为：

set_layer_rc -layer M1 -capacitance 0.0001 -resistance 0.001

对于 Via，set_layer_rc 只需要电阻值（单位 kOhm）：

- Via Resistance: 1.00 ohm = 0.001 kOhm

set_layer_rc -via VIA1 -resistance 0.001

请为所有相关的金属层生成完整的 setRC.tcl 文件。

3 运行 PhyLS

本章介绍如何配置并执行 PhyLS 工具。

3.1 命令行参数

PhyLS 使用 JSON 文件来指定所有运行配置。

命令语法：

```
./phyls config.json
```

参数说明：

参数	默认值	描述
--designs	/designs	**(必需)** 设计文件目录的绝对或相对路径（参见 2.2.2 设计目录）。
--platform	/platform	**(必需)** 平台文件 (PDK) 所在目录的绝对或相对路径（参见 2.2.1 平台 (PDK) 目录）。
--design_name	ac97_top	**(必需)** 设计的顶层模块名称，必须与文件基本名称一致。
--temp_dir	.	临时工作目录路径，用于存放中间文件。运行结束后自动清理。
--output_verilog	final.v	输出的优化 Verilog 网表文件名。
--mode	timing	优化模式。timing：时序驱动优化（默认）；area：面积驱动优化。详见 3.2 优化模式。
--effort	mid	优化力度。high：最高质量，无迭代上限；mid：平衡模式（默认）；low：最快收敛。详见 3.4 优化力度。仅在 Timing 模式下生效。

参数	默认值	描述
--is_floorplan	False	启用 Floorplan 模式，从 floorplan DEF 重建时序驱动布局。详见 3.3 工作模式说明。
--output_def	False	PhyLS 在增量布局后输出 DEF 文件。
--def_file	final.def	DEF 文件名（仅在 --output_def 启用时使用）。
--output_spef	False	启用 SPEF 输出，用于寄生参数估计。
--spef_file	phyls.spef	SPEF 文件名（仅在 --output_spef 启用时使用）。
--thread_count	1	PhyLS 并行处理的线程数，建议设置为 16 或 32，不超过 CPU 核心数。
--hier	False	启用层次化 link_design 模式。适用于包含层次化模块实例的设计。
--verbose	False	启用详细输出，在每次算子执行后打印关键路径报告。
--timing_budget_factor	1.5	仅 Area 模式有效。允许的时序退化因子（默认 1.5 = 允许 TNS/WNS 退化 50%）。详见 3.2.2 Area 模式。

3.2 工作模式说明

PhyLS 根据输入 DEF 文件的类型支持两种工作模式，通过 --is_floorplan 参数选择。

3.3.1 Floorplan 模式（使用 --is_floorplan）

使用场景： 输入 DEF 是 floorplan，仅定义了芯片面积和标准单元行，**没有标准单元布局**。

适用于： 早期阶段优化，从 floorplan 开始生成高质量的初始布局。

3.3.2 Placed 模式（默认，不使用 --is_floorplan）

使用场景： 输入 DEF 已包含标准单元的布局信息。

适用于： 对已布局设计的增量优化和细化。

3.3 优化力度（Effort）

PhyLS Timing 模式提供三种优化力度等级，通过 --effort 参数选择。

Effort	最大迭代数	无改进容忍次数	描述
high	60	10	最高质量
mid	30	6	平衡模式（默认）。适用于大多数场景。
low	15	3	最快收敛。

注： Area 模式不支持 --effort 参数，使用固定的迭代配置。

3.4 推荐使用流程

PhyLS 推荐使用**两阶段优化流程**以获得最佳结果：

阶段 1：Floorplan 到布局（使用 --is_floorplan + --output_def）

目的： 从 floorplan DEF 开始，通过 PhyLS 的 AI 驱动布局引擎生成优化的布局，作为后续细化的高质量初始解。

输入：

- Floorplan DEF：包含芯片边界、IO 位置、宏单元，但无标准单元布局
- 综合后的 Verilog 网表
- 时序约束

输出：

- 优化的 Verilog 网表
- **已布局的 DEF 文件**（通过 --output_def 生成）

关键参数：

- **必须使用** --is_floorplan 标志
- **必须使用** --output_def 标志
- 指定 --def_file 输出文件名

阶段 2：增量布局细化（不使用 --is_floorplan，不输出 DEF）

目的： 对阶段 1 生成的已布局 DEF 进行增量优化。由于输入已经是布局后的 DEF，**无需再输出 DEF**。

输入：

- 阶段 1 输出的已布局 DEF（复制到设计目录，替换原 floorplan DEF）
- 阶段 1 输出的优化 Verilog 网表（或更新后的网表）

输出：

- 最终优化的 Verilog 网表

关键参数：

- **不使用** --is_floorplan 标志（默认 Placed 模式）
- **不使用** --output_def 标志（无需输出 DEF）
- 可使用 --output_spef 生成寄生参数文件

---

4 输出与报告

本章描述 PhyLS 工具成功运行后生成的输出文件和报告。

4.1 工具输出

4.1.1 优化网表 (Verilog)

- **文件：** 由 --output_verilog 参数指定（默认：final.v）
- **描述：**
  - 这是 PhyLS 对输入网表进行时序、功耗和面积优化的结果。
  - 此输出网表确保与输入网表**逻辑等价**。
  - 该网表已为后续的 P&R 步骤做好了优化，实现更快的时序收敛。

4.1.2 布局文件 (DEF)

- **文件：** 由 --def_file 参数指定（默认：final.def）
- **生成条件：** 仅在使用 --output_def 标志时生成
- **描述：**
  - 包含经过增量全局布局优化后的单元位置信息
  - 包括所有标准单元、宏单元的坐标和朝向
  - 可用作后续 P&R 流程的高质量初始布局
- **使用场景：**
  - **推荐**在阶段 1（Floorplan 模式）中输出
  - **不推荐**在阶段 2（Placed 模式）中输出（因为输入已是布局后的）

4.1.3 寄生参数文件 (SPEF)

- **文件：** 由 --spef_file 参数指定（默认：phyls.spef）
- **生成条件：** 仅在使用 --output_spef 标志时生成
- **描述：**
  - 标准 SPEF 格式的寄生参数提取文件
  - 包含基于布局的互连 RC 寄生信息
  - 用于下游的精确时序分析和签核
- **使用场景：**
  - **推荐**在阶段 2（最终优化）中输出
  - 可导入 PrimeTime、Tempus 等 STA 工具进行 signoff

4.2 QoR 报告

PhyLS 在运行过程中输出全面的 QoR 指标。

Timing 模式输出指标（每次迭代）：

指标	单位	描述
WNS	ns	最差负裕量 (Worst Negative Slack)
TNS	ns	总负裕量 (Total Negative Slack)
Leakage power	uW	静态漏电功耗
Slew violations	ns	总 slew 违例余量
Capacitance violations	fF	总电容违例余量

Area 模式额外输出（运行结束时）：

指标	单位	描述
Initial area	um^2	初始标准单元面积
Best area	um^2	优化后的最佳面积
Area reduction	%	面积缩减百分比

日志还包含每次迭代的耗时分解：

- Scan Time：门级违例扫描耗时
- Operator Time：算子执行耗时
- Evaluate Time：设计评估耗时
- Report Time：报告生成耗时

附录 A 7nm GPU 模块实测案例

本章节展示了 PhyLS 在某 GPU 公司 7nm 工艺核心模块上的实测结果。

A.1 测试背景

- 工艺节点：7nm FinFET
- 设计规模：~96 万门
- Baseline：Cadence Full Flow (Genus + Innovus)
- 对比流程：Baseline + PhyLS (Genus + PhyLS + Innovus)

A.2 结果对比

在相同的物理约束和时序约束下，PhyLS 流程展现出了显著的时序修复能力，特别是针对大规模的时序违例路径（TNS 和 NVP）有极大的改善。

Metric	Baseline	PhyLS flow	Improvement
WNS (ns)	-0.245	-0.237	+3.06%
TNS (ns)	-501.972	-254.925	+49.22%
违例路径数量 NVP (Sum)	15201	9283	+38.93%
Density	54.88%	52.99%	+3.44%
Total Power (mW)	948.15	937.66	+1.11%
Runtime (h)	27.20	26.99	+0.79%
PeakMem (GB)	47.58	46.22	+2.85%

A.3 结论

在此 7nm GPU 模块中，PhyLS 在保持 WNS 基本持平的情况下，成功消除了**超过 50% 的总时序违例（TNS）**和**违例路径数量（NVP）**。这不仅大幅降低了后端 P&R 工具的修复压力，还有效减少了后续 ECO 迭代的次数，同时实现了约 2.3% 的功耗节省。