Received 00 April, 2024; revised 11 April, 2024; accepted 00 April, 2024; Date of publication 00 May, 2024; date of current version 24 June, 2024.

Digital Object Identifier 10.1109/DD.2024.0624000

Descriptor: *Physical Design Database (PDB)*

Yunxiang Zhang[1,*], Kian Kit Cheah[1,2,*], Fu Qi Chua[1,2], Yuhang Zhang[1], Harikrishnan Ramiah[2], Huimin Wen[1,†], and Yongfu Li[1,†]

[1]School of Integrated Circuits, Shanghai Jiao Tong University, Shanghai, China.
[2]Department of Electrical Engineering, University of Malaya, Malaysia.

CORRESPONDING AUTHORS: Yongfu Li (email: yongfu.li@sjtu.edu.cn) and Huimin Wen (e-mail: wenhuimin@sjtu.edu.cn).

Y. Zhang and K. Cheah contributed equally to this article.
This work is supported in part by the National Natural Science Foundation of China under Grants No. 62350610271 and in part by the Explorers Program of Shanghai (Basic Research Funding) under Grant No. 24TS140020.

ABSTRACT The Physical Design Database (PDB) introduces a collection of physical circuit designs and associated metadata to advance optimization research on the chip development process. This comprehensive dataset comprises various circuit prototypes, ranging in complexity and scale, and spans multiple technology process nodes, enabling diverse use cases and applications. Each entry in the database consists of a physical layout in widely used GDSII or DEF formats, along with detailed metadata that covers instance, verification, and performance-power-area (PPA) metrics. The PDB also provides a suite of tools, including an automated layout generator, for efficiently generating layouts and metadata directly from Register Transistor Level (RTL) designs. This resource is intended to facilitate performance optimization, assist in developing layout generation tools, and serve as a benchmark for analyzing and evaluating physical layouts. By providing a uniform structure for layout data, the PDB enables researchers to optimize circuits more effectively, supporting the development of more efficient layout generation methodologies and promoting advances in semiconductor design and manufacturing.

IEEE SOCIETY/COUNCIL Circuits and Systems Society (CASS)

DATA DOI/PID 10.21227/s9j2-cw80

DATA TYPE/LOCATION Text; Shanghai, China

INDEX TERMS physical design, layout, performance-power-area (PPA) metric, semiconductor dataset.

Background

THE rapid advancement of semiconductor technology has led to significant progress in high-performance chips used in fields such as autonomous driving [1], high-performance computing [2], [3], portable devices [4], and AI computing [2]. As chips achieve higher integration levels and greater design complexity, the demand for optimized performance-power-area (PPA) metrics has grown significantly, with critical factors including clock frequency, latency, thermal management, and area efficiency [3]. These advancements require a more refined chip development process to meet increasingly stringent PPA requirements [5].

The chip development process incorporates multistage feedback verification loops, as illustrated in Fig. 1. This process ensures both manufacturability and compliance with target metrics, guaranteeing successful tape-out and production. In this process, the physical layout of a circuit refers to the detailed geometric representation and spatial arrangement of a chip's devices, components, interconnections, and routing layers. It determines the precise placement, orientation, and wiring at the physical implementation level, directly influencing critical metrics such as signal integrity, timing characteristics, power consumption, manufacturability, and overall reliability [6]. Hence, it serves as a critical bridge between the chip design flow and the chip manufacturing

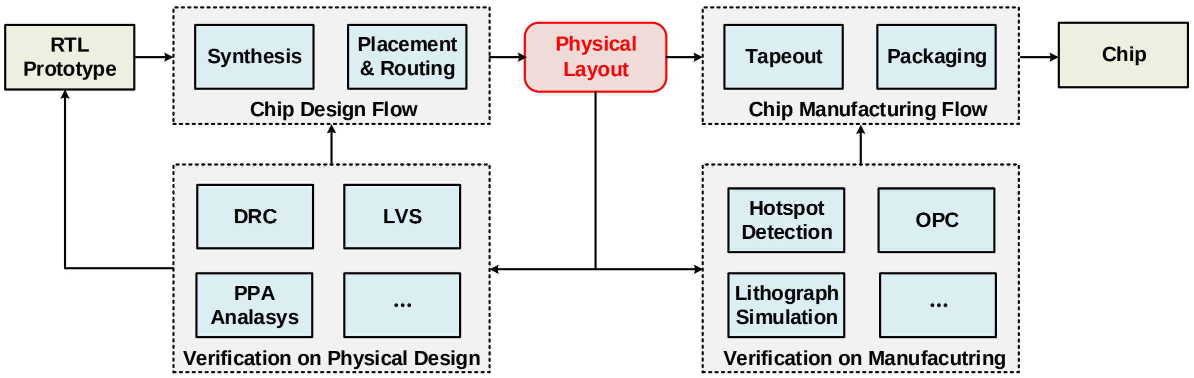


Fig. 1. The basic flow of the chip development process.

flow, supporting various verification stages within the chip development process.

To achieve these objectives, it is insufficient to express the chip's physical placement and orientation in a standalone layout file. Detailed and systematic records of the layout generation process, along with layout metrics, are also essential as metadata for the layout. With these records, we can accurately observe and assess the quality of a layout and improve the chip generation process. For example, by tracking the PPA metrics of a layout, we can optimize circuit prototypes. By analyzing changes in these performance metrics throughout the layout generation process, we can guide improvements in Electronic Design Automation (EDA) tools used in the chip development process [7].

Comprehensive layout datasets, including layout files, generation flow records, and metric records, are rare in the open-source community. The available open-source layout data, including those from open-source routing contests (e.g., ISPD18 [8] and ISPD19 [9]) and standard circuit benchmarks (e.g., ISCAS85 and ISCAS89) [10], only provides final layout files and lacks critical information about generation records and evaluation metrics. These layout data are suitable for geometry-based research, such as physical verification [11], serving as testing datasets; however, they do not effectively support improvements in the chip development process due to the absence of these records.

To address the scarcity of open-source physical design data with comprehensive records, this work presents the Physical Design Database (PDB), which utilizes OpenROAD [12] and its auxiliary project, OpenROAD-Flow-Scripts (ORFS) [13], to generate a series of physical designs. The PDB provides a collection of physical layout models derived from various open-source circuits, using multiple technology process nodes to generate layouts. By systematically aggregating diverse physical layout implementations with comprehensive metadata and generating reports, the PDB establishes a framework for capturing layout generation workflows and associated layout metrics. Furthermore, the project includes an ORFS-based automated layout generation tool, which enables users to convert custom Register Transistor Level (RTL) circuit prototypes into physical layouts while systematically collecting corresponding records to rebuild or expand the database. With these capabilities, the PDB establishes comprehensive traceability throughout the entire layout generation flow, while simultaneously enabling data-driven optimization of the chip development process. Unlike previous datasets, PDB combines RTL-level netlists, physical layouts, and complete flow metadata in a single framework. This integration not only subsumes the functionality of both RTL and layout collections but also makes the database directly applicable to benchmarking and optimization of EDA tools and layout generation flows.

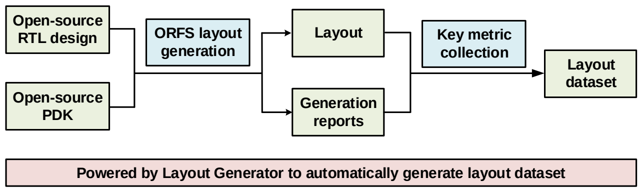


Fig. 2. The data collection flow of PDB.

Collection Methods and Design

This section details the automated data collection pipeline used to create our PDB dataset. As shown in Fig. 2, our automated pipeline uses the ORFS layout generation workflow to produce a physical layout and generation reports from an open-source RTL design. Key metrics are then extracted from the layout and combined with the reports to create an entry in the final Layout dataset. This entire process is powered by our Layout Generator tool, which enables users to generate layouts and collect metadata from their own custom circuit designs.

Source of the Layout Database

The RTL circuit prototypes are primarily sourced from open-source models [14], [15]. These models encompass fundamental RTL circuit designs such as arithmetic circuit units, processors, and image encoding circuits, among other common hardware architectures. We collect a diverse range

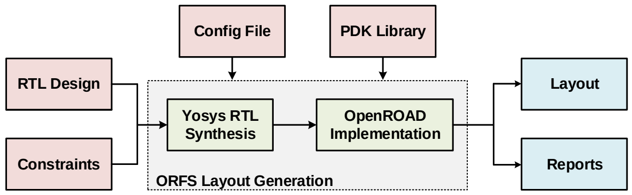


Fig. 3. The flow of layout generation by OpenROAD-Flow-Scripts.

of circuits to generate layouts to enhance the coverage of circuit types.

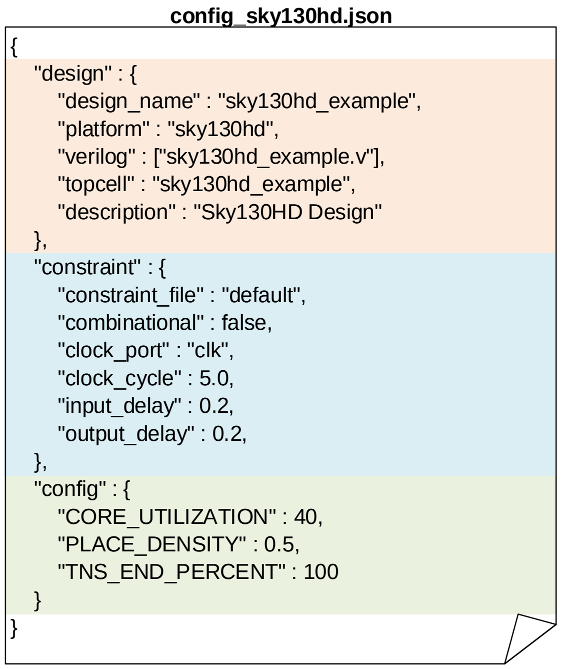
Process Design Kits (PDKs) are also essential for the layout generation flow. A PDK typically includes information such as technology-specific design rules, device models, and parameterized cells, which are crucial for determining layout metrics. For the PDB, the Nangate45 PDK is mainly employed [16] (with 142 layouts). We also utilize Sky130HD [17] (with 133 layouts) and GF180 [18] (with 129 layouts) as our PDK libraries to enhance the coverage of technology nodes.

Layout Generation Flow

The layout generation workflow utilizing OpenROAD Flow Scripts (ORFS) is depicted in Fig. 3. Generating layouts from RTL designs comprises two primary stages: RTL synthesis and implementation. Within this workflow, RTL synthesis is predominantly performed using Yosys, an open-source synthesis tool [19]. Yosys transforms the RTL design into a netlist file, which specifies the interconnections among standard cells. The implementation stage then utilizes this netlist to generate the physical layout through a sequence of steps, including floorplan, placement, clock tree synthesis, and routing.

To automatically execute the layout generation flow by ORFS, we developed a specialized tool named Layout Generator. This tool automates the generation of physical layouts from RTL source code and concurrently collects associated metadata, ultimately forming a structured dataset. The workflow requires users to supply RTL source files and essential design constraints through a JavaScript Object Notation (JSON) configuration file, which enables the generation of physical layouts.

As shown in Fig. 4, the JSON configuration file integral to the Layout Generator is structured into three primary sections: Design, Constraint, and Config. The Design section describes the specification of fundamental design parameters, including the design name, PDK name, and top-level cell name. The Constraint section allows users to define timing constraints pertinent to the circuit type (i.e., combinational or sequential). These may include clock periods and input/output delays, or users can provide a custom Standard Design Constraints (SDC) file. The Config section is used to specify physical parameters for the layout, such as core area utilization and placement density. It also permits adjustments to


config_sky130hd.json

```
{
  "design" : {
    "design_name" : "sky130hd_example",
    "platform" : "sky130hd",
    "verilog" : ["sky130hd_example.v"],
    "topcell" : "sky130hd_example",
    "description" : "Sky130HD Design"
  },
  "constraint" : {
    "constraint_file" : "default",
    "combinational" : false,
    "clock_port" : "clk",
    "clock_cycle" : 5.0,
    "input_delay" : 0.2,
    "output_delay" : 0.2,
  },
  "config" : {
    "CORE_UTILIZATION" : 40,
    "PLACE_DENSITY" : 0.5,
    "TNS_END_PERCENT" : 100
  }
}
```

Fig. 4. The structure of the configuration JSON file of the Layout Generator.

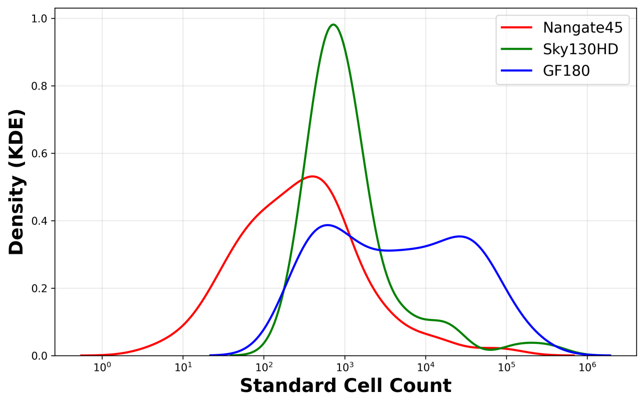
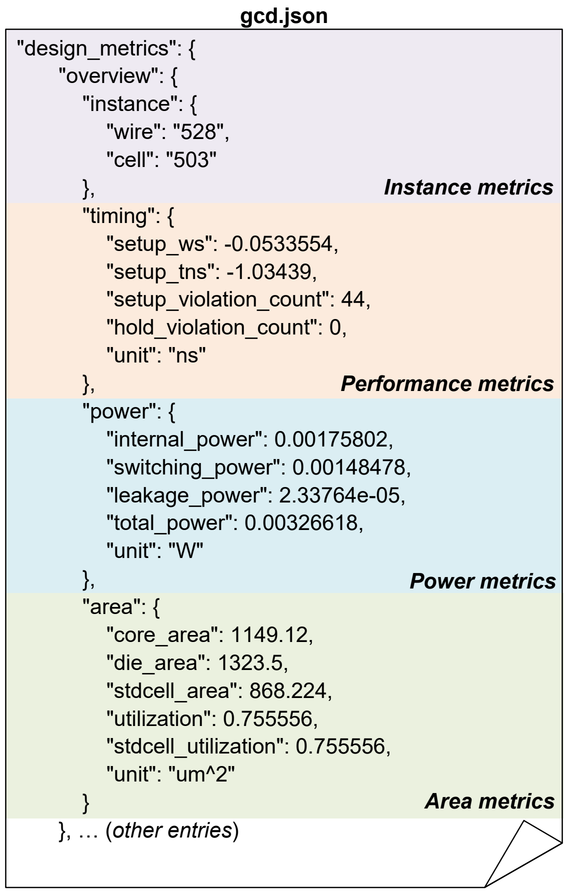


Fig. 5. Kernel Density Estimation (KDE) of standard cell distribution across different PDKs.

various parameters to optimize the layout generation process. An accompanying Makefile script streamlines the execution of the Layout Generator, utilizing both user-provided RTL source files and the comprehensive JSON configuration to drive the automated workflow.

Layout Collection Flow

The layouts generated by ORFS from the provided RTL source files are systematically collected in the PDB. Fig. 5 presents a Kernel Density Estimation (KDE) plot that visualizes the distribution of standard cell counts across three different Process Design Kits (PDKs). This distribution reveals the typical design scale for each technology, indicating its most likely application scenarios. The plot shows that Nangate45 and Sky130HD predominantly feature


gcd.json

```
"design_metrics": {
    "overview": {
        "instance": {
            "wire": "528",
            "cell": "503"
        },
        "timing": {
            "setup_ws": -0.0533554,
            "setup_tns": -1.03439,
            "setup_violation_count": 44,
            "hold_violation_count": 0,
            "unit": "ns"
        },
        "power": {
            "internal_power": 0.00175802,
            "switching_power": 0.00148478,
            "leakage_power": 2.33764e-05,
            "total_power": 0.00326618,
            "unit": "W"
        },
        "area": {
            "core_area": 1149.12,
            "die_area": 1323.5,
            "stdcell_area": 868.224,
            "utilization": 0.755556,
            "stdcell_utilization": 0.755556,
            "unit": "um^2"
        }
    }, ... (other entries)
```

Fig. 6. The overview of design metrics in Greatest Common Divisor (GCD) JSON file.

“small-scale” circuits (10-1000 cells), making them well-suited for basic functional verification and rapid prototyping, with Sky130HD designs clustering sharply at the upper end of this range. In contrast, the GF180 PDK’s bimodal distribution highlights its versatility, accommodating both these small-scale applications and larger “medium-to-large-scale” circuits ($>10^4$ cells) intended for comprehensive performance and robustness evaluations.

In addition to the physical layout data, metadata records are systematically collected to describe each layout’s pertinent physical and electrical metrics. To comprehensively summarize the essential characteristics of a design, instance metrics are utilized alongside PPA metrics as shown in Fig. 6. These metrics are typically extracted from reports generated by the RTL-GDSII tool during the layout generation process. Besides, the evolution of these metrics throughout each subprocess of the layout generation flow is also recorded. Analyzing the variations in these metadata at different stages enables an assessment of the efficacy of the physical design tools and methodologies employed in each subprocess.

Instance metrics focus on the layout’s dimensions and the utilization of standard cells. The layout size is characterized by the number of cells and nets, which indicates the design’s complexity. Standard cell coverage quantifies the proportion of unique standard cell types from the PDK that are instantiated in the layout relative to the total number of available standard cell types in the PDK, thereby reflecting the diversity of cells employed.

Performance metrics primarily assess the adherence to the circuit’s timing requirements. Specifically, the longest signal delay path, or the critical path, is analyzed to determine compliance with timing constraints. The difference between the maximum permissible delay for this critical path and its actual propagation delay is referred to as timing slack. A negative timing slack, indicative of a timing violation, signifies that signal propagation delays on this path may impede correct signal transmission, potentially leading to functional errors. Key performance indicators include the worst slack (WS), total negative slack (TNS), and the total count of timing violations within the circuit [20].

Power metrics encompass dynamic and static power consumption. Dynamic power is further categorized into internal power, which is the power consumed by internal nodes during signal transitions synchronized with the clock, and switching power, which is the power dissipated at primary output nodes during signal changes. Static power primarily accounts for power consumption due to leakage currents [20].

Area metrics include the standard cell area, core area (which incorporates routing area), and die area (which consists of the area for input/output pads and power distribution). These area metrics are crucial for analysis related to routing congestion, voltage (IR) drop, and thermal management [20].

The Layout Generator can also automatically archive key intermediate files generated by ORFS and output the generated physical layout, along with the aforementioned layout metadata, to create a layout dataset.

Validation and Quality

To ensure that our layout database has sufficient reference value, we verify the reliability using the following methods: (i) Design Rules Check (DRC) and Layout Versus Schematic (LVS) checks on each physical layout by KLayout to ensure the physical correctness and consistency between the circuit and layout. (ii) Cross-generation verification of the circuits using industry-standard layout generation tools, comparing the key metrics of the results generated by two different process flows to show the comparability. (iii) The capability and robustness of the custom layout generation by Layout Generator to ensure that Layout Generator can accept various user-specific circuit prototypes to generate layouts.

DRC and LVS check with KLayout

Fig. 7 shows the KLayout verification flows of DRC and LVS. The DRC process verifies the layout’s geometric structures and spacing against predefined rules. There are two input files for KLayout DRC: a physical layout file and a

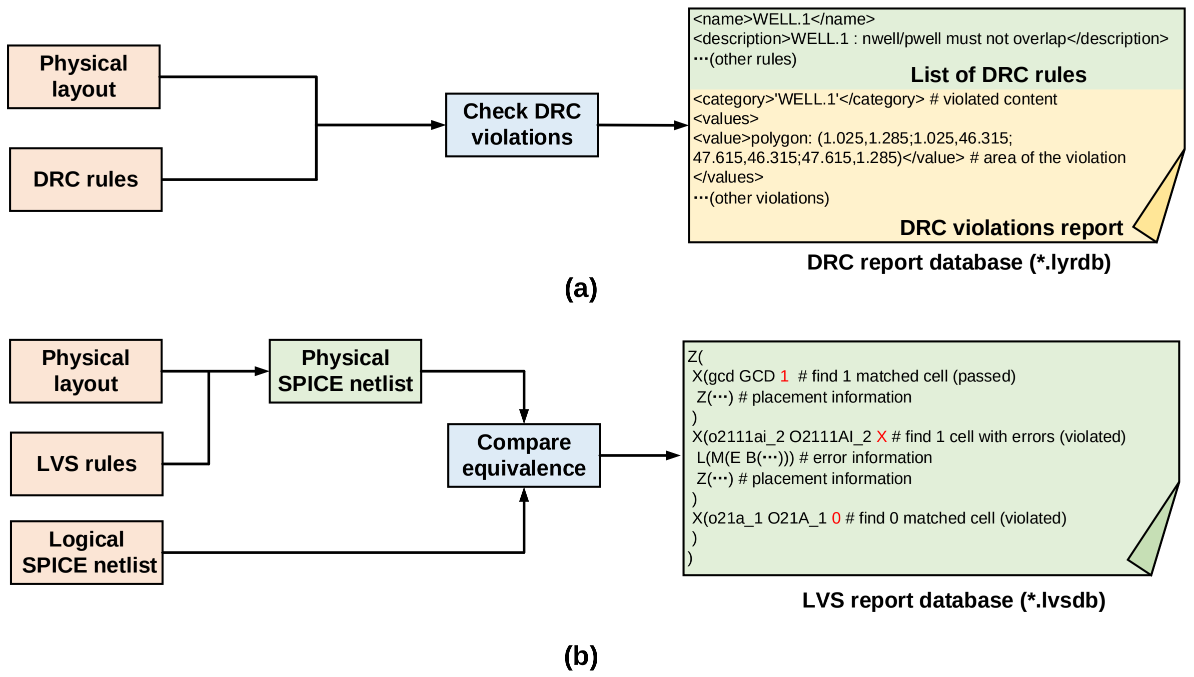

Physical layout
DRC rules
Check DRC violations
<name>WELL.1</name>
<description>WELL.1 : nwell/pwell must not overlap</description>
···(other rules)
List of DRC rules
<category>'WELL.1'</category> # violated content
<values>
<value>polygon: (1.025,1.285;1.025,46.315;
47.615,46.315;47.615,1.285)</value> # area of the violation
</values>
···(other violations)
DRC violations report
DRC report database (*.lyrdb)
(a)
Physical layout
LVS rules
Logical SPICE netlist
Physical SPICE netlist
Compare equivalence
Z(
X(gcd GCD 1 # find 1 matched cell (passed)
Z(···) # placement information
)
X(o2111ai_2 O2111AI_2 X # find 1 cell with errors (violated)
L(M(E B(···))) # error information
Z(···) # placement information
)
X(o21a_1 O21A_1 0 # find 0 matched cell (violated)
)
)
LVS report database (*.lvsdb)
(b)

Fig. 7. The KLayout verification flows of (a) DRC and (b) LVS.

DRC rules file. After running the check, KLayout generates a DRC verification database file (`.lyrdb`) that includes all the DRC rules and the DRC violation information. The DRC check ensures the physical manufacturability across all layers of the layout, thereby avoiding fabrication issues and guaranteeing reliable, defect-free manufacturing.

The LVS check in KLayout verifies the consistency between the layout and the schematic by comparing the physical SPICE netlist generated from the layout with the logical SPICE netlist. KLayout LVS takes the layout, logical SPICE netlist, and LVS rule files as inputs. In the LVS flow, ORFS uses the KLayout tool to translate the physical design information into a SPICE netlist, which is then compared with the logical SPICE netlist generated by logic synthesis using the LVS rule file. After verification, KLayout produces an LVS database file (.lvsdb) that records results across the entire circuit by checking the consistency of each standard cell instance. This ensures that the layout accurately corresponds to the schematic and implements the intended functionality.

Our database covers DRC/LVS validation across three open-source PDKs (two commercial and one educational PDK). A small subset of layouts was unable to be validated due to the inability to recognize some external cells that do not belong to the PDK, such as FakeBRAMs and DFFRAMs, without SPICE netlist models being available. As summarized in Table I, most layouts can pass DRC and LVS checks. Our database provides reliable and reproducible physical verification support by integrating rule files, repairing libraries, and systematically collecting verification results.

TABLE I. Summary of DRC and LVS validation coverage across different PDKs.

PDK	Source of Layouts	# of Designs	DRC # of Pass	LVS # of Pass	LVS # of Fail[1]
Nangate45	CORE	123	123	123	0
	ORFS	19	19	10	9
Sky130HD	CORE	123	123	123	0
	ORFS	10	10	8	2
GF180	CORE	123	123	123	0
	ORFS	6	6	4	2

[1]External cells that do not belong to the PDK's IP library, such as FakeBRAMs, which do not provide SPICE netlist models, so they will not be able to create a complete/clean LVS run.

Cross-Validation Using Industry-standard Layout Generators

Cross-generational verification was performed to assess the technical quality of the dataset. This involved generating layouts for a representative subset of the input circuits using recognized industry-standard layout generation tools. These tools used the same input RTL and design constraints as those used to generate the corresponding layouts within the PDB. Comparative results for layout key metrics for the Advanced Encryption Standard (AES) and Greatest Common Divisor (GCD) circuit designs are presented in Tables II and III, respectively. Analysis of the data presented in these

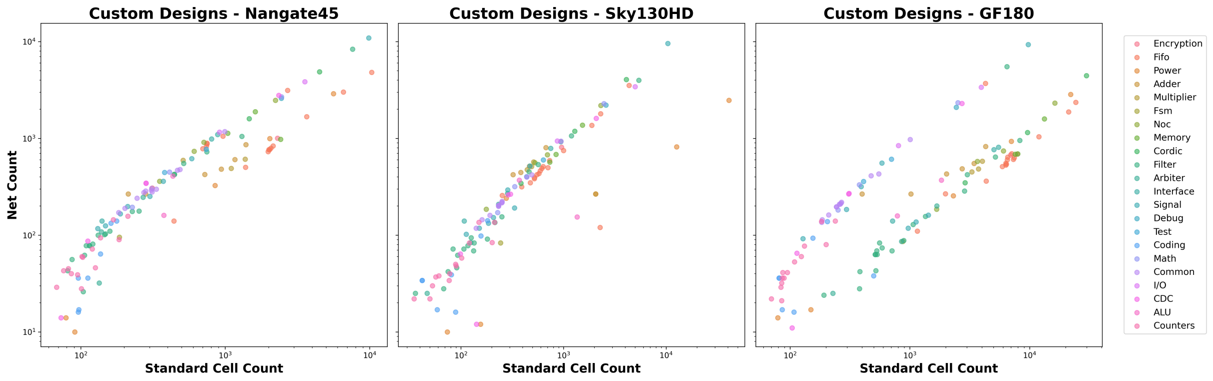


Fig. 8. The generation result of custom designs by the Layout Generator.

TABLE II. Comparison of AES's PPA metrics with ORFS and Commercial EDA tools.

Metrics		ORFS	Commercial
PDK		Nangate45	
clock cycle/ns		0.82	
floorplan utilization		predefined floorplan DEF	
standard cell count		15057	13838
Performance	worst slack/ns	-0.124	-0.004
	total negative slack/ns	-7.92	-0.15
	violation count	155	63
Power	internal power/W	0.249	1.18e-02
	switching power/W	0.255	1.28e-02
	leakage power/W	8.09e-04	3.26e-04
	total power/W	0.505	2.49e-02
Area	die area/μm^2	62612.4	62612.41
	core area/μm^2	52785	52785.04
	standard cell area/μm^2	28202.4	14323.568

TABLE III. Comparison of GCD's PPA metrics with ORFS and Commercial EDA tools.

Metrics		ORFS	Commercial
PDK		Nangate45	
clock cycle/ns		0.46	
floorplan utilization		0.55	
standard cell count		503	409
Performance	worst slack/ns	-0.053	-0.14
	total negative slack/ns	-1.034	-5.16
	violation count	44	47
Power	internal power/W	1.76e-03	7.85e-04
	switching power/W	1.48e-03	4.45e-04
	leakage power/W	2.38e-05	1.14e-05
	total power/W	3.27e-03	1.24e-03
Area	die area/μm^2	1323.5	795.239
	core area/μm^2	1149.12	790.44
	standard cell area/μm^2	868.224	521.094

tables indicates that the layout metrics for designs generated using the PDB's ORFS-based flow are comparable to those achieved with the benchmarked industry-standard tools.

Custom Layout Generations by Layout Generator

To validate the operational reliability and versatility of the Layout Generator, its capability to process diverse, user-defined circuit designs was evaluated. We selected CORE [15] as the custom circuit design testcase due to its broad coverage of various circuit types. The suite of circuits included standard digital logic modules, such as Arithmetic Logic Units (ALUs), Adders, First-In-First-Out (FIFO) buffers, Networks on Chip (NoCs), Counters, and Finite State Machines (FSMs). The results of these layout generation experiments using custom designs are summarized in Fig. 8. These outcomes demonstrate the proficiency of the Layout Generator in interpreting various user-defined RTL prototypes and a wide range of technology nodes and layout sizes, effectively producing their corresponding physical layouts and highlighting the robustness of the generation process.

Records and Storage

Here, we outline the file storage structure of our database to facilitate users in utilizing our layout data and layout generation tools for layout generation and analysis. Additionally, we provide a detailed description of the layout metric record file for each layout, enabling users to extract key metrics from this file.

Overview

As shown in Fig. 9, the PDB comprises curated layout datasets (`/layout`), an automatic layout generation framework (`/layout_generator`), and supporting scripts for analysis (`/scripts`) and documentation of the PDB and the Layout Generator. The modular directory structure enables explicit annotation between design data, generation tools, and evaluation resources.

The dataset's primary records, comprising individual physical design entries, are organized within design-specific subfolders under the top-level `/layout/{PDK}` directory. Each design-specific folder contains a consistent set of

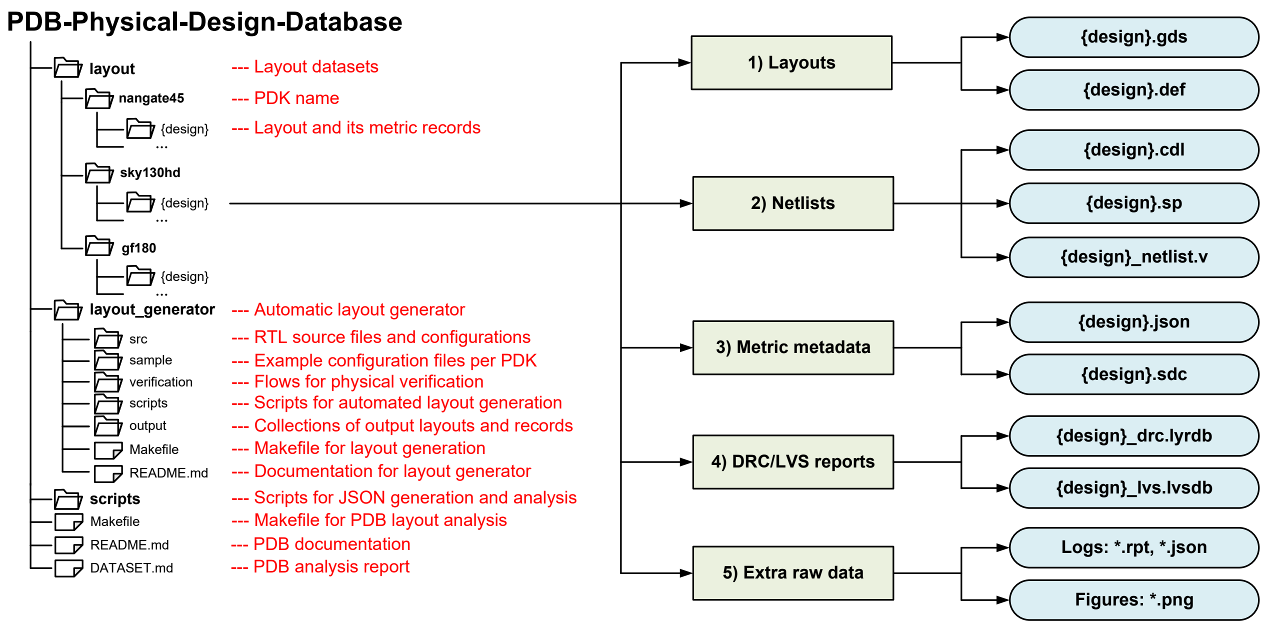

PDB-Physical-Design-Database
layout --- Layout datasets
nangate45 --- PDK name
{design} --- Layout and its metric records
sky130hd
{design}
gf180
{design}
layout_generator --- Automatic layout generator
src --- RTL source files and configurations
sample --- Example configuration files per PDK
verification --- Flows for physical verification
scripts --- Scripts for automated layout generation
output --- Collections of output layouts and records
Makefile --- Makefile for layout generation
README.md --- Documentation for layout generator
scripts --- Scripts for JSON generation and analysis
Makefile --- Makefile for PDB layout analysis
README.md --- PDB documentation
DATASET.md --- PDB analysis report
1) Layouts
{design}.gds
{design}.def
2) Netlists
{design}.cdl
{design}.sp
{design}_netlist.v
3) Metric metadata
{design}.json
{design}.sdc
4) DRC/LVS reports
{design}_drc.lyrdb
{design}_lvs.lvsdb
5) Extra raw data
Logs: *.rpt, *.json
Figures: *.png

Fig. 9. **The file structure of the PDB.**

files and subfolders encompassing the design outputs and associated metadata. Key components typically include:

1) **Layout Files**: Two common layout formats are provided: GDSII (files with a ".gds" extension) and DEF (files with a ".def" extension).
2) **Netlist Files**: Provided netlist files include Verilog netlists generated from logic synthesis (e.g., `{design}_netlist.v`), corresponding logical SPICE netlists (e.g., `{design}.cdl`), and physical SPICE netlists (e.g., `{design}.sp`) generated by KLayout LVS.
3) **Layout Metadata Files**: Each circuit design is accompanied by a comprehensive metadata file in JSON format (e.g., `{design}.json`). This file contains essential design parameters, key performance metrics, and references to verification reports. This metadata is collected from various reports and logs generated throughout the automated layout process. Additionally, an SDC file (e.g., `{design}.sdc`) defines the timing constraints of the circuit, including clock definitions, input/output delays, and specific timing exceptions such as false paths and multicycle paths as a supplement to the timing metrics.
4) **DRC and LVS Verification Reports**: Design Rule Check (DRC) and LVS reports, generated using KLayout, are provided. These reports validate the physical correctness of the layout and its consistency with the original circuit schematic, thereby supporting the technical quality and reference value of the dataset.
5) **Supplementary Raw Data**: Intermediate logs, report files, and JSON data records originating from the ORFS layout generation process are supplied as supplementary raw data. These files document the layout generation workflow in the `/logs` subfolder. Additionally, diagrams visualizing abstract layout structures (e.g., clock trees, placement maps, and routing diagrams), generated by ORFS, are available in the `/figures` subfolder.

Beyond the `/layout` directory containing the described design entries, the `/layout_generator` directory contains the source files for the Layout Generator tool. The structure of the Layout Generator is detailed below:

1) `/src` houses custom circuit RTL designs and other relevant configuration files.
2) `/sample` includes example JSON configuration files intended for use with the Layout Generator tool. These files also serve as templates, providing default values for parameters that might be absent in user-supplied configurations for the generator.
3) `/verification` contains technology files, scripts, and rule decks for KLayout used in the design verification workflow.
4) `/output` is the default location for storing physical design data generated when users run the Layout Generator tool with their circuit designs. Furthermore, these generated outputs can be installed in the `/layout` directory, becoming part of the PDB.

The key scripts located in the `/scripts` are detailed below:

1) `flatten.lydrc`: A KLayout script utilized to generate flattened GDSII layout files from the initial design data. Users can directly manipulate the geometries

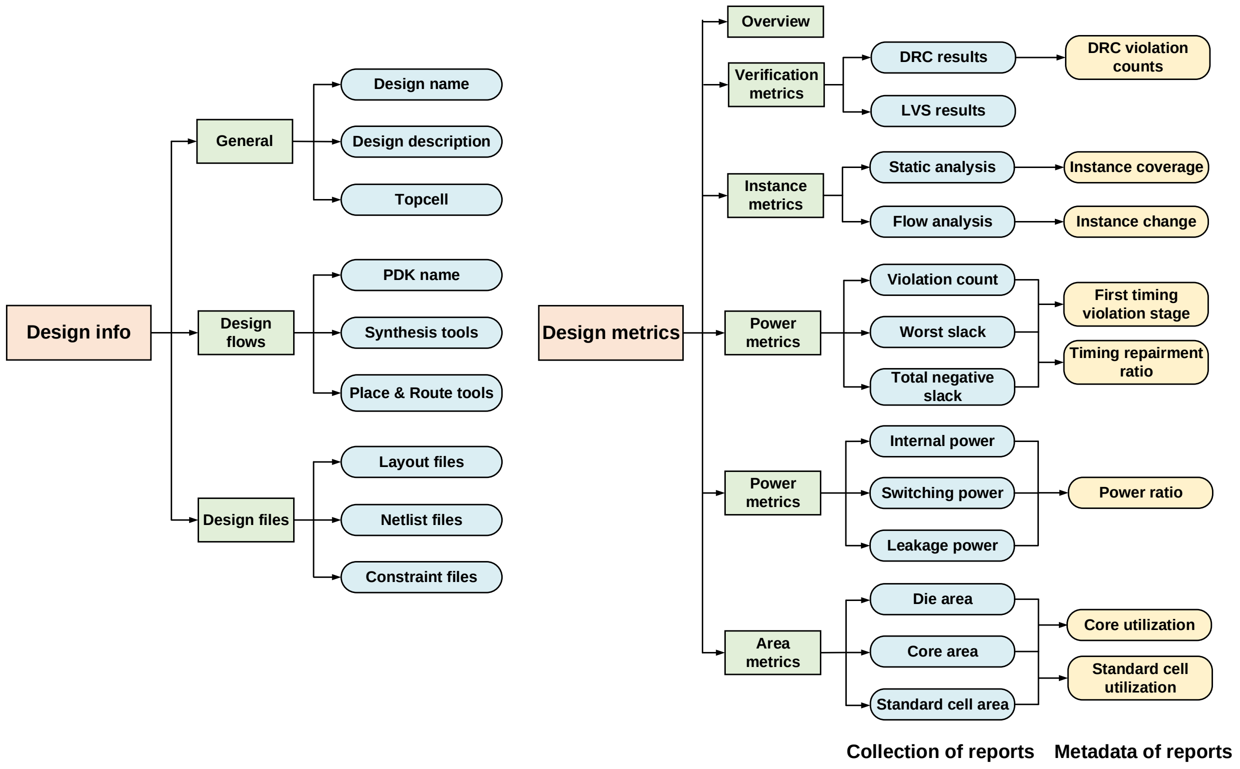


Fig. 10. The structure of JSON metadata for a layout, including 2 sections: (a) “Design info” and (b) “Design metrics”.

in the flattened layout for related research on layout graphics.

2) `generate_json.py`: This Python script parses outputs from design tools and layout files to extract and compile key metadata. It generates structured JSON files for each physical design in the PDB, forming a core component of the queryable database.
3) `analyze_data.py`: A Python utility script for performing various PDB analysis and visualizations. Its functionalities include tracking and visualizing changes in metadata for individual layouts throughout their different generation stages, as well as generating graphical representations of the distribution, range, and coverage of standard cell counts and types across the entire layout database. The outputs from this analysis script, including key visualizations and summary data, are documented in the `DATASET.md` file, located in the repository's root directory on GitHub.

Structure of Layout Metadata

As illustrated in Fig. 10, the metadata for each design is structured as a JSON record in the file `{design}.json` for each layout. It comprises two primary keys: **design_info** and **design_metrics**:

“design_info”: This key provides essential details for each design, including its name, top-level cell identifier, a brief description, the PDK, design tools, and direct references to the corresponding netlist and layout files. This information facilitates the straightforward identification and retrieval of core design files.

“design_metrics”: This key comprehensively aggregates key design characteristics and performance indicators. It encompasses the following sub-sections:

1) **“overview”**: Contains a summary of fundamental layout metrics, including instance counts (e.g., number of standard cells and nets) and consolidated PPA values. These data provide a concise snapshot of the layout's primary characteristics.
2) **“verification_metrics”**: Includes references to physical verification report files (e.g., DRC and LVS reports) and a summary of critical violation statistics.
3) **“instance_metrics”**: Furnishes a detailed structural analysis of various instance types, such as standard cells, fill cells, tap cells, and vias, including their counts and spatial distributions. This analysis is presented in two categories:

(a) **Static Quantitative Analysis**: Details the counts of standard cells in both the logical netlist and the final physical layout. The metadata also includes an analysis of the standard cell coverage ratio, which represents the proportion of utilized cells from the complete Process Design Kit (PDK) library. It highlights any standard cells from the PDK that are not underrepresented in the design.

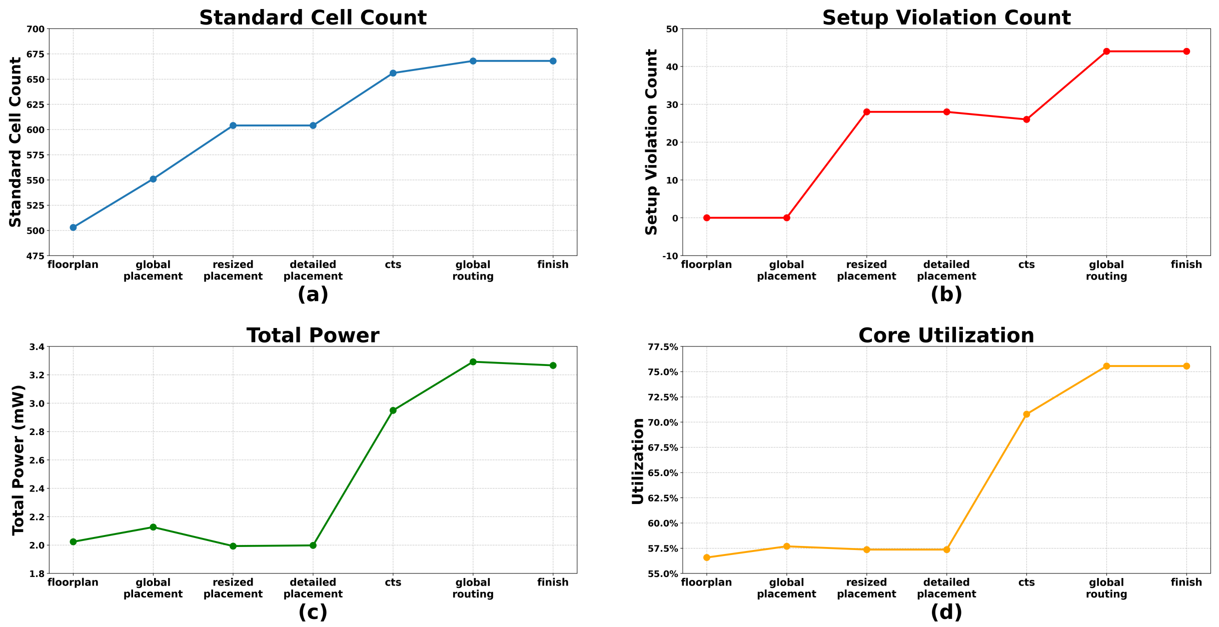


Fig. 11. Design flow metrics of the GCD circuit: (a) Standard cell count, (b) Setup violation count, (c) Power consumption, and (d) Core utilization across each stage.

(b) **Dynamic Flow Analysis**: Documents the counts of standard cells and total instances across distinct stages of the design process. A report detailing the net changes for each standard cell type between each subprocess is also provided.

4) **PPA metrics**: This section is further organized into three sub-keys corresponding to PPA in the JSON metadata record: **"timing_metrics"**, **"power_metrics"**, and **"area_metrics"**. These entries document variations in key PPA parameters discussed in Section II across different design stages. Analysis of these PPA metrics has been included to enhance data quality, such as timing repair ratios for "timing_metrics", the ratio of each part of power for "power_metrics", and core utilization percentages for "area_metrics".

Insights and Notes

Here, we detail key applications of the layout database, illustrating how the accompanying metadata extends its utility for diverse research and development activities.

Circuit Design and RTL-GDSII Tool Optimization

The chip development process requires comprehensive data to optimize design metrics and improve performance. The PDB provides a detailed overview of the layout generation process and key metrics, facilitating circuit design enhancement and the optimization of RTL-GDSII tools. In circuit design, using layout generation tools to refine chip metrics is essential [21], [22]. To support this, the PDB dataset includes benchmark PPA values for each circuit, offering a reliable performance reference for circuit prototypes within ORFS. Researchers can leverage these benchmarks as a foundation for optimizing parameters in circuit design and for enhancing layout generation methodologies. As shown in Fig. 11, an illustrative example of the changes in instance count and key PPA metrics for the layout generation of a Greatest Common Divisor (GCD) circuit is provided. This comprehensive metadata enables users to track metric variations throughout the layout generation process, identify critical stages, and apply targeted optimizations.

Layout Database for Physical Verification

The physical verification process in chip back-end design relies on various algorithms, including DRC, LVS, Multiple Patterning Lithography Decomposition, pattern matching, and hotspot detection. These algorithms typically require pre-generated physical layouts as input data, as demonstrated in several recent studies that utilize such datasets for algorithm development and testing [11], [23]–[25]. The PDB provides a curated collection of physical layouts, enabling researchers to directly utilize them for algorithm development and testing, thus accelerating the research process.

Additionally, for research focused on the geometric structures of layouts, such as MPLD [11] and pattern matching [25], we offer a script to flatten the hierarchical layout structure into Manhattan polygons. For studies on cell coverage and utilization, such as DRC rules coverage [26] and physical testing coverage [27], we provide detailed reports on cell usage frequency and coverage in the JSON metadata included with the dataset. As shown in Fig. 12, our dataset allows users to analyze the extent to which each cell from the provided PDK library is used in various physical design instances. It also identifies unused cells in each layout,

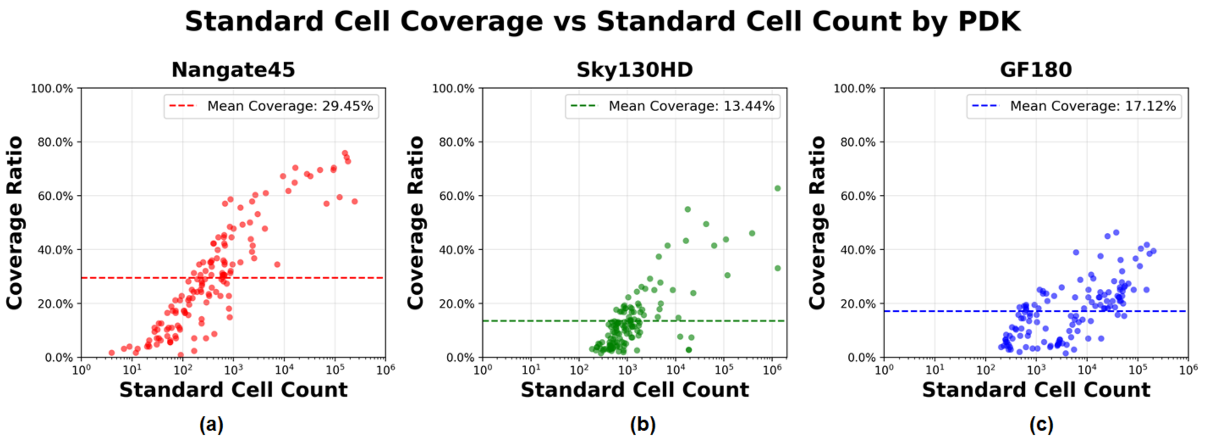


Fig. 12. **Standard cell coverage vs. standard cell counts on different PDKs: (a) Nangate45, (b) Sky130HD, and (c) GF180.**

enabling a more informed selection of layout combinations to maximize cell coverage for more comprehensive algorithm testing and benchmarking.

Source Code and Scripts

For transparency and reproducibility, all layout data and the scripts used in the data generation of our PDB have been available on GitHub: https://github.com/SJTU-YONGFU-RESEARCH-GRP/PDB-Physical-Design-Database. It is also available on IEEE DataPort: https://ieee-dataport.org/documents/physical-design-database-pdb.

The generation flow of layouts based on OpenROAD-Flow-Scripts [28], with Yosys 0.47+121 [29] as its synthesis tool and OpenROAD v2.0-17198-g8396d0866 [30] as its placement and routing tool. The various custom digital circuit prototypes are from CORE [15]. KLayout 0.29.8 [31] is used for visualizing layouts and performing physical verification.

Acknowledgements and Interests

Y. Zhang, K. Cheah, F. Chua, and Y. Li are the main authors of this work. Y. Zhang, K. Cheah, H. Wen, and Y. Li designed the data collection system, performed data cleaning, and prepared the data for publication. All authors reviewed the manuscript.

This work is supported in part by the National Natural Science Foundation of China under Grants No. 62350610271 and in part by the Shanghai Explorer Program under Grant No. 24TS140020.

The authors of the article have declared no conflicts of interest.

REFERENCES

[1] O. J. Arndt, P. Rallapalli, and H. Blume, "Chapter 6 - Portable implementations for heterogeneous hardware platforms in autonomous driving systems," in *Big Data Analytics for Cyber-Physical Systems*, G. Dartmann, H. Song, and A. Schmeink, Eds. Elsevier, 2019, pp. 113–143. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128166376000063
[2] J. R. Hu, J. Chen *et al.*, "Systematic co-optimization from chip design, process technology to systems for GPU AI chip," in *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2018, pp. 1–2.
[3] V. Saravanan, G. Sasikala *et al.*, "Innovative Techniques in the Construction of Very Large Scale Integration Circuits for Low-Power and High-Performance Circuits," in *Global Conference on Communications and Information Technologies (GCCIT)*, 2024, pp. 1–6.
[4] D. Basak, M. H. Miraz *et al.*, "Analysis of a comparator for low delay, high speed and compact area for portable device application," in *International Conference on Digital Applications, Transformation Economy (ICDATE)*, 2023, pp. 1–6.
[5] N. Hanchate, "Improving Chip Design Performance and Productivity Using Machine Learning," *Proceedings of the International Symposium on Physical Design*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248151095
[6] R. Reis, F. L. Kastensmidt, and J. L. Güntzel, "Physical design methodologies for performance predictability and manufacturability," in *Proceedings of the Conference on Computing Frontiers*, ser. CF '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 390–397. [Online]. Available: https://doi.org/10.1145/977091.977147
[7] W. R. Davis, P. Franzon *et al.*, "Fast and Accurate PPA Modeling with Transfer Learning," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–8.
[8] I. . C. Organizers, "Ispd 2018 initial detailed routing contest benchmarks," 2018, accessed: 2025-06-14. [Online]. Available: https://www.ispd.cc/contests/18/#benchmarks
[9] ——, "Ispd 2019 initial detailed routing contest benchmarks," 2019, accessed: 2025-06-14. [Online]. Available: https://www.ispd.cc/contests/19/#benchmarks
[10] Y. Ma, "Iscas benchmark suite," 2020, accessed: 2025-06-14. [Online]. Available: https://yuzhema.people.ust.hk/release/ISCAS_benchmark.zip
[11] F. Yu, J. Shen *et al.*, "MAED: Mask assignment encoder decoder solver for multiple patterning layout decomposition," *Expert Systems with Applications*, vol. 268, p. 126309, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424031762
[12] T. Ajayi, T. Spyrou, and A. Kahng, "The OpenROAD project: Unleashing hardware innovation," *OpenROAD Project*, 2021. [Online]. Available: https://doi.org/10.1145/3316781.3326334
[13] A. S. Udani, S. Roy, and U. Barman, "Chip design using OpenROAD," in *International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, 2024. [Online]. Available: https://doi.org/10.1109/ICEECT61758.2024.10739008
[14] O. Community, "Opencores: Open source hardware community," https://opencores.org/, 2025, accessed: 2025-05-21.
[15] K. K. Cheah, F. Q. Chua *et al.*, "Descriptor: A Corpus of Synthesizable Verilog RTL Modules Dataset for EDA Research (CORE)," *IEEE Data Descriptions (DD)*, Dec 2025.

[16] K. Cheung, "What is the first open source 45nm standard-cell library?" *SIGDA Newsletter*, vol. 38, no. 8, Apr. 2008. [Online]. Available: https://doi.org/10.1145/1862870.1862871

[17] Google, "Skywater open pdk (sky130hd)," https://github.com/google/skywater-pdk, 2021, accessed: 2025-06-06.

[18] Google, "GlobalFoundries 180nm MCU PDK (gf180)," https://github.com/google/gf180mcu-pdk, 2021, accessed: 2025-06-06.

[19] C. Wolf, J. Glaser, and J. Kepler, "Yosys-A Free Verilog Synthesis Suite," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:202611483

[20] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.

[21] H. Kuang and L. Wang, "Multi-objective Design Space Exploration for High-Level Synthesis via Bayesian Optimization," in *International Symposium of Electronics Design Automation (ISEDA)*, 2023, pp. 150–155.

[22] S. Saha, A. Khatri *et al.*, "Physical Synthesis Optimization Prediction Using Machine Learning," in *2025 38th International Conference on VLSI Design (VLSID)*, 2025, pp. 25–30.

[23] Z. Lin, Z. Gu *et al.*, "Hotspot Detection with Machine Learning Based on Pixel-Based Feature Extraction," *Scientific Programming*, vol. 2022, pp. 1–11, 08 2022.

[24] C.-L. WU and C.-W. LU, "Pattern Matching for Feasible and Efficient Physical Design Verification of Cell Libraries," *IEICE Transactions on Electronics*, vol. E108.C, no. 1, pp. 34–45, 2025.

[25] W. Ge and C. Wang, "Efficient Layout Pattern Matching Based on Local Information," in *IEEE International Conference on ASIC (ASICON)*, 2023, pp. 1–4.

[26] Y. Lee, J. Park *et al.*, "DRC code coverage test a novel QA methodology," in *International Conference on IC Design Technology (ICICDT)*, 2018, pp. 93–96.

[27] Y. Nagamura, K. Shiozawa *et al.*, "Layout-based test coverage verification for high-reliability devices," in *International Symposium on Semiconductor Manufacturing (ISSM)*, 2016, pp. 1–4.

[28] T. O. P. contributors, "Openroad-flow-scripts: Open-source rtl-to-gds flow," https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts, 2023, accessed: 2025-05-26.

[29] C. Wolf and Y. contributors, "Yosys open synthesis suite," https://github.com/YosysHQ/yosys, 2023, accessed: 2025-05-26.

[30] T. O. P. contributors, "The openroad project," https://github.com/The-OpenROAD-Project, 2023, accessed: 2025-05-26.

[31] M. Köfferlein and K. contributors, "Klayout - layout viewer and editor," https://github.com/KLayout/klayout, 2023, accessed: 2025-05-26.