iPD: An Open-source intelligent Physical Design Toolchain

(Invited Paper)

Xingquan Li[1,3*], Simin Tao[1], Shijian Chen[1,2], Zhisheng Zeng[1,2], Zhipeng Huang[1], Hongxi Wu[4,3], Weiguo Li[5,3], Zengrong Huang[1], Liwei Ni[1,2], Xueyan Zhao[2,10,3], He Liu[6,1], Shuaiying Long[1], Ruizhi Liu[2,10,3], Xiaoze Lin[1,2], Bo Yang[1,8], Fuxing Huang[4,3], Zonglin Yang[7,3], Yihang Qiu[10,3], Zheqing Shao[9,3], Jikang Liu[7,3], Yuyao Liang[7,3], Biwei Xie[2,1,✉], Yungang Bao[2,10,3,1,✉], and Bei Yu[11,✉]

[1]Peng Cheng Laboratory, [2]Institute of Computing Technology, Chinese Academy of Sciences, [3]Beijing Institute of Open Source Chip, [4]Fuzhou University, [5]Minnan Normal University, [6]Peking University, [7]Shenzhen University, [8]Sun Yat-sen University, [9]University of Science and Technology of China, [10]University of Chinese Academy of Sciences, [11]The Chinese University of Hong Kong
Email: *lixq01@pcl.ac.cn, ✉xiebiwei@ict.ac.cn, ✉baoyg@ict.ac.cn, ✉byu@cse.cuhk.edu.hk

2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC) | 979-8-3503-9354-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASP-DAC58780.2024.10473932

***Abstract*—Open-source electronic design automation (EDA) shows promising potential in unleashing EDA innovation and lowering the cost of chip design. The open-source EDA toolchain is a comprehensive set of software tools designed to facilitate the design, analysis, and verification of electronic circuits and systems. We developed a physical design EDA toolchain (named iPD) from netlist to GDS-II, including design, analysis, and verification. iPD now covers the whole flow of physical design (including floorplan, placement, clock tree synthesis, routing, timing optimization etc.), part of the analysis tools (timing analysis and power analysis), and part of the verification tools (design rule check). For more friendly support EDA research and development and chip design, we design a reliability, extendibility, ease-of-use, and feature richness physical design toolchain. This paper introduces the software structure, functions, and metrics of the iPD toolchain.**

I. Introduction

The rapid advancement of digital technology has led to a substantial increase in demand for the growth of the integrated circuit (IC) industry. However, Moore's law is reaching its limits [1]. To meet the growing needs for computing and storage, we may explore alternative integration technologies, such as 3D chip design or package and chiplets. Moreover, we should strive to improve the quality of chip design, especially in the area of EDA. EDA tool and design methodology have been the subject of research for many years. Fortunately, new techniques such as artificial intelligence (AI), hardware acceleration, and collaborative design of chips and EDA have demonstrated their capacity and practical value in EDA. These advancements open up the potential for further optimizing EDA tools and methodology.

To maximize the benefits of new techniques in EDA, we need open-source EDA tools with accessible to all EDA enthusiasts. The desirable open-source EDA tools should be automated, well-designed, and compatible while supporting tapeout while offering high performance and scalability, providing comprehensive documentation, and easy-to-learn. In the pursuit of designing high-quality open-source EDA tools, there have been numerous valuable explorations and contributions in the past. Many researchers and developers have dedicated their efforts to advancing the field and creating useful tools. These tools are classified according to the integrated circuit design process, including simulation, logic synthesis, formal, physical design and sign-off, etc. Existing open-source tools can be combined into a design process from RTL to GDSII. Nowadays, there are already several open-source digital physical design tools, such as EDA tools in OpenROAD [2], DREAMPlace [3], Xplace [4], Ripple [5], Rsyn [6], graywolf [7], Qrouter [8] CUGR [9], Dr.CU 2.0 [10], and OpenTimer [11]. And they work well to promote open collaboration and innovation in the EDA community.

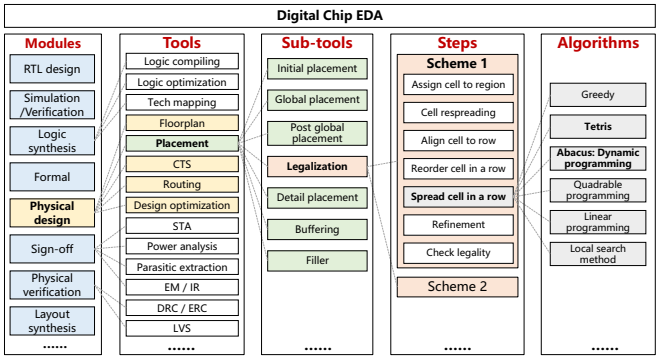


Fig. 1: EDA decomposition.

In summary, open-source EDA tools are currently in a stage of diverse opinions, with each project having its own characteristics and gradually becoming more established. They cover the most important tools and can meet chip design requirements. However, most open-source EDA tools are developed and maintained by professors and students in academia. Initially, the code for some tools often comes from research papers. Most existing open-source EDA tools only support one algorithmic approach. To be compatible with a wider range of chip design goals and requirements, an EDA tool system must support various solutions and algorithms. Additionally, existing open-source EDA tools face challenges in achieving optimal reliability, extendibility, ease-of-use, and feature richness. These situations discourage contributors and users from actively participating in open-source EDA projects, resulting in a lack of diverse contributors.

This work focuses on designing an open-source intelligent physical design EDA toolchain (iPD)[1]. Besides supporting chip

[1]https://github.com/OSCC-Project/iEDA/tree/master/src/operation

design from netlist to GDS-II, a more important objective of iPD is to provide a EDA tool research and development (R&D) platform for EDA practitioners (researchers, EDA designers, algorithm developers, and software engineers, etc). To support various algorithmic solutions, the EDA tool platform needs to be highly extendable and have a solid EDA foundation and evaluation system. Efforts are being made to improve the quality and reliability of open-source EDA tools. This includes enhancing documentation, establishing community-driven development processes, promoting industry collaboration, and providing long-term support. The iPD project aims to attract diverse academic disciplines, foster collaboration, and bridge the gap between industry and academia, facilitating the development of valuable EDA design methodologies. Currently, the iPD toolchain can support chip implementation from netlist to GDS-II for block-level chips and system-on-chips (SoCs) at the 28nm process node. It offers a reliable, highly extendable, and user-friendly physical toolchain, breaking through basic core technologies. iPD consists of nine physical design and analysis EDA tools designed through a decomposition and integration way.

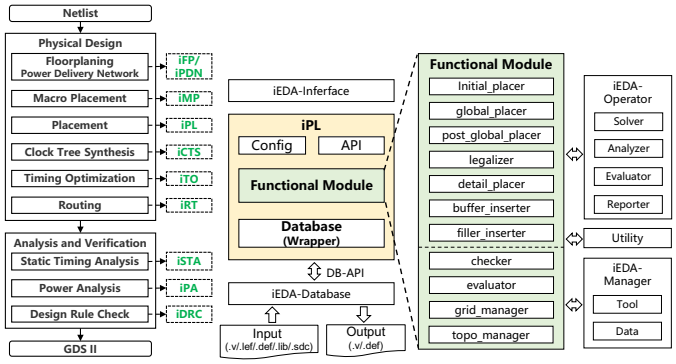
II. Decomposition, Integration and Software

The physical design is a complex process that involves multiple objectives and constraints to be optimized. It includes assessing electrical objectives such as timing, power consumption, noise, signal and power integrity (SI/PI), and electromigration; checking design rules, electrical rules, manufacturability, and reliability requirements. Especially for modern chip designs that require features of process nodes, physical design has become extremely challenging and is considered the most difficult step in the entire chip design process. Therefore, to ensure feasibility and optimization, traditional physical design is typically divided into multiple stages and steps.

A. Decomposition and Problem

To design a robust and comprehensive physical design toolchain, we first perform a hierarchical decomposition of the physical design module as shown in Fig. 1. The physical design module can be decomposed into multiple tools, including floorplan, placement, cts, routing, and design optimization; Each of these tools can be further decomposed into multiple sub-tools, such as placement, which can be further divided into initial placement, global placement, post-global placement, legalization, detailed placement, buffering, and filler; Each subtool can be further decomposed into several key steps to implement, such as the following steps to achieve legalization: aligning cells to a valid region, spreading cells within the region, aligning cells to rows, rearranging cells within rows, spreading cells within rows, refinement, and checking legality; For each step, there are multiple valuable algorithms, such as modeling the spreading of cells within rows. With the different data representation, spreading of cells within rows can be modeled as dynamic programming, quadratic programming or linear programming and so on. Under different test cases, these algorithms will have different performance and advantages.

Through such tool decomposition, we can decompose the EDA system into multiple modules and tools, hundreds of steps,



(a) (b)

Fig. 2: (a) Chip design steps and iPD tools; (b) The software structure of iPL (the same for others).

and thousands of algorithmic problems. To address the key challenges, we can abstractly describe the critical problems and form a white paper on EDA key problems. By collaborating with scholars from multiple disciplines, we can jointly tackle these challenges. For example, some problems are listed as follows: How can we obtain more accurate metric evaluation in macro placement? How can we establish a differentiable timing, power and congestion model integrated in placement? How can we achieve good coordination between clock tree synthesis and placement and routing? How can we quickly generate a good Steiner tree or routing result on weighted layout? How can we quickly solve large-scale high-order interconnect delay model?

B. Solution and Integration

For complex EDA systems, a basic consensus is that no algorithm can perform best for any case. This is because each algorithm has its own advantages and limitations, and the performance of the algorithm varies in different application scenarios and conditions. To obtain a tool that supports various functional extensions and various algorithms. The design philosophy of the tools in the iPD toolchain mainly consists of two principles. Firstly, each tool is designed as a hierarchical set of subtools, steps, and algorithms. Secondly, multiple algorithms are supported for each key technology.

The design process of iPD can be summarized as follows: firstly, each tool is decomposed into subtools and steps; then several valuable solutions are designed to address the key problems; then these solutions are integrated to form feasible algorithms for each step, multiple steps are integrated to form subtools, and subtools are combined to form the basic version of the tool. For different design requirements, functional extensions can be made at different levels.

C. Software Architecture

iPD is designed to construct open-source EDA research tools and algorithm sets, with the aim of designing more extensible EDA tools at minimal cost. We adopt a decoupled design structure, with key design requirements, including: firstly, designing based on the open-source EDA infrastructure iEDA to reduce development costs; secondly, separating data and operational functions within the tool; thirdly, the tool being a hierarchical algorithm set, embedded in the overall design chain through an

plug-and-play approach; and fourthly, all tools in iPD having the same structure.

An example tool structure of placement (named iPL) is shown in Fig. 2(b), which utilizes iEDA's database, operator, manager, and interface to organize data and algorithms. The iPL mainly includes the iPL-database and functional modules. Users can configure tool functions and parameter flows through the config files, and then obtain corresponding functional outputs through the API.

III. iPD: Physical Design Toolchain

iPD aims to support the flow of chip design from Netlist design to GDS-II layout. The chip design steps and the iPD tools are shown in Fig. 2(a). Each tool is composed of several low-coupling functional operations, which work by calling a series of different algorithms on designated data models.

A. Floorplan and Power Delivery Network

As the initial stage of chip physical design, floorpaln aims to determine chip area, meet the requirements of routing, and ensure timing closure and chip stability.

Functions. In the iPD toolchain, floorplan is implemented by iFP and iPDN, including the following main functions: layout initialization, I/O planning, physical cell placement and power delivery network (PDN) generation. iFP and iPDN provide various of interfaces to support initializing design process files (including tech lef file and verilog file), planning IO ports, pads, tap cell, endcap cell and blocks, generating connected PDN wires and vias for all layers. iFP and iPDN also offer some useful APIs to accelerate chip flow development. iFP and iPDN can show floorplan result as layout, can present data summary report which statistics some typical data such as core utilization, net numbers, PDN distribution in different layer and so on.

Implemented Key Technologies. In iFP, we achieve distance-based cluster and hierarchical cluster methods to group IO cells and pins according to instance connection and some user defined requirements.

B. Macro Placement

Macro placement serves as the initial stage in the placement process, provides opportunities for design optimization. Nonetheless, macro placement presents challenges due to potential deviation in metrics evaluation, as well as a vast solution space with multiple objectives and constraints, making the exploration process considerably daunting. At present, conventional chip macro placement relies on engineers' expertise and decision-making.

Functions. iMP is an automated tool for macro placement that aids engineers in reducing design time, optimizing various targets, and enhancing macro placement quality. Our iMP has several functions, including determining the location of macros, planning the region of standard cells, and satisfying constraints such as blockage, guidance, and so on. At present, the evaluation metric employed for iMP is the half-perimeter wirelength (HPWL) between macros and clusters.

Implemented Key Technologies. To achieve a better macro placement result, the main techniques utilized in iMP include a simulated annealing (SA) algorithm that employs sequence pair representation, and a non-linear programming method based on Poisson equation [12]. Since netlist partition or cluster in physical design are valuable solvers, we implement hMetis partition and multilevel cluster in iMP.

C. Standard Cell Placement

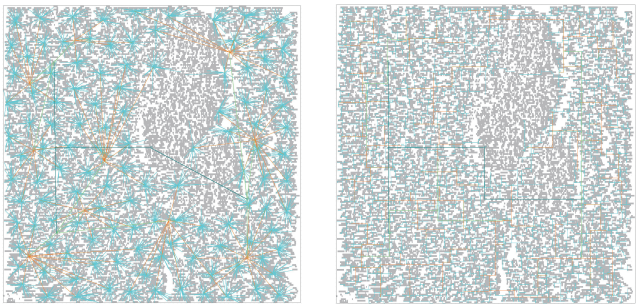
Placement mainly ensures the proper coordinate of each cell mapped in the netlist within a designated region, which must comply with design rules and be conducive to routing, timing convergence, and power consumption.

Functions. iPL is mainly composed of initial placer, global placer, post-global placer, legalizer, detail placer, filler, buffer inserter and checker. The objective of iPL is to minimize wirelength, timing and congestion. iPL supports reading and writing layout-related data from the iEDA data source and supports user-defined placement parameters, can easily use iEDA infrastructure interface. iPL constructs topology manager and grid manager for netlist and placement area management. The topology manager is responsible for extracting and managing hypergraph information. iPL uses it to construct wire length models such as HPWL, Steiner tree WL (STWL), and weighted average WL (WAWL), as well as the RC-tree required by iSTA. The grid manager manages layout information, including the management of cell distribution area on bin and the management of the legality check site. iPL prepares for timing optimization and routability optimization. iPL supports calling iSTA to obtain timing information of placement status, supports calling the early global routing to obtain routing information, supports incremental legalization, and supports inserting buffers to optimize wire length and timing. iPL supports netlist information printing as well as layout timing and pre-routing information reporting.

Implemented Key Technologies. For the placement process, iPL uses the electric field density model and WAWL model for global placement of cells, uses the Abacus and Tetris algorithm with the minimum movement target for cell legalization, and finally uses greedy algorithms such as in-line cell shift, global cell exchange, and local reordering. In the initial and global placement stages of iPL, we have achieved quadratic programming, conjugate gradient method and Nesterov optimization method and non-uniform fast Fourier transform. In legalization and detail placement, we implement dynamic programming and minimum-cost flow method. iPL supports timing optimization, uses timing path endpoint information dissemination to construct net/path weights, and uses net-weighted global placement iteration and search window relocation to optimize timing indicators. With the congestion evaluation by Rudy metric and calling iRT, iPL supports routability optimization. Based on the pre-routing information of the evaluated layout area Bin, global cell expansion and network flow movable cell methods are used to eliminate congestion.

D. Clock Tree Synthesis

Clock tree synthesis (CTS) aims to deal with clock net and to balance skew among flip-flops while optimizing design resource usage, under timing constraints. It facilitates the creation of



(a) (b)

Fig. 3: iCTS results: (a) Clock net topology; (b) Clock net routing result.

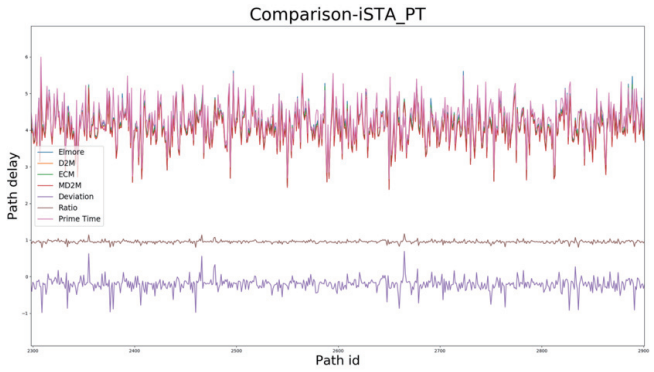


Fig. 4: Timing regression between iSTA and PT.

topologies with diverse objectives through the utilization of multiple merging criteria.

Functions. iCTS has a built-in timing calculation model and utilizes robust clustering methods to detect violations in nets, allowing for the reassignment of cells and nets. iCTS supports the construction of buffering solutions using buffers with different sizes. It provides comprehensive layout and routing results during the clock tree construction process. It also offers optimization for clock nets, including fanout, wirelength, and capacitance. iCTS enables the creation of clock trees with lower latency and controllable skew. It supports various topology generation strategies, such as greedy distance, greedy merge cost, bi-partition, and bi-cluster. Additionally, iCTS supports a hierarchical architecture, allowing for the specification of different constraint schemes at different levels. The iCTS provides generation of latency and skew reports through interaction with iSTA. It also provides statistics on inserted buffers and wirelength. An illustrative clock tree result of iCTS is shown in Fig. 3.

Implemented Key Technologies. In terms of timing, we have implemented a cell-driving capability machine learning model to estimate cell delay that effectively guides users in design constraints. By incorporating a lower bound estimation of cell delay, we address the hysteresis issue that arises from buffering in the clock tree merging process. For clustering methods, we have developed multiple focused clustering techniques for CTS, such as K-means cluster, distance cluster and min-cost flow. Additionally, we have optimized the violation metrics by utilizing simulated annealing (SA). To construct a desirable clock tree, we have designed the balanced skew latency tree (BEAT) based on the deferred-merge embedding (DME) framework [13]. We have also incorporated the Steiner shallow-light tree (SALT) approach [14]. BEAT ensures a balance between routability, hierarchical design, and various constraints, resulting in superior performance and design robustness while utilizing fewer resources.

E. Routing

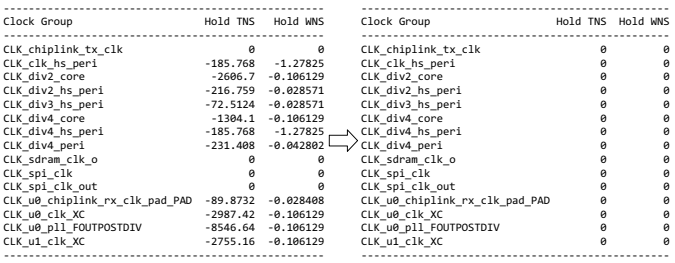
Routing involves the physical interconnection between embedded components within a chip. Our routing tool is called iRT, which aims to meet performance and functional requirements while considering factors such as design rule check (DRC) and signal integrity. iRT supports constructing data from iEDA database, and its main modules include pin accessor, resource allocator, global router, track assigner, detailed router, and violation repairer. Additionally, it also provides interfaces for early global router, timing evaluation, and net physicalization.

Functions. From the perspective of the routing flow, firstly, the pin accessor can find the best accessible pin points. To coordinate routing resource among nets and reduce the negative effect of net order, iRT supports routing resource pre-assignment, i.e, it will assign appropriate resources for each net before routing. Next, the global router combines the congestion map and resource map (from the resource allocator) to perform routing on a three-dimensional grid using the routing algorithm. iRT can carefully evaluate and control overflow through rip-up and reroute, and it can adaptively choose the routing layer. iRT pre-route long wire at track assignment stage by considering the routability of subsequent detailed routing. At detail routing, iRT optimizes design rules by calling a built-in DRC engine and evaluates timing by calling the APIs of iSTA. To obtain better routing results, the violation repairer step of iRT will eliminate local routing violations. iRT considers wirelength and number of vias as basic objectives. It utilizes iDRC to count design rule violations and employs iSTA to evaluate the timing of the routing.

Implemented Key Technologies. To establish legal access points for pins, we achieve static and dynamic conflict elimination by constructing a conflict graph at the pin accessor step. The resource allocator is modeled as a quadratic programming (QP) and is solved by using fast gradient descent to obtain a resource map for each net. The global router considers finer-grained resource control and uses a fast Steiner tree generation (SLUTE) algorithm and routing algorithms (including pattern routing, dynamic routing and multiple sources and multiple sinks A* routing) to account for potential overflow in subsequent stages. At the track assigner, we consider non-preferential measures to increase routing space and extend the endpoint to the pin for detailed routing. The detailed router iteratively eliminates a large number of design rule violations by performing 3D A* algorithm.

F. Static Timing Analysis

iSTA is a static timing analysis (STA) tool that provides easy-to-use interfaces for accessing required timing data. iSTA supports



Clock Group	Hold TNS	Hold WNS
CLK_chiplink_tx_clk	0	0
CLK_clk_hs_peri	-185.768	-1.27825
CLK_div2_core	-2606.7	-0.106129
CLK_div2_hs_peri	-216.759	-0.028571
CLK_div3_hs_peri	-72.5124	-0.028571
CLK_div4_core	-1304.1	-0.106129
CLK_div4_hs_peri	-185.768	-1.27825
CLK_div4_peri	-231.408	-0.042802
CLK_sdram_clk_o	0	0
CLK_spi_clk	0	0
CLK_spi_clk_out	0	0
CLK_u0_chiplink_rx_clk_pad_PAD	-89.8732	-0.028408
CLK_u0_clk_XC	-2987.42	-0.106129
CLK_u0_pll_FOUTPOSTDIV	-8546.64	-0.106129
CLK_u1_clk_XC	-2755.16	-0.106129

Clock Group	Hold TNS	Hold WNS
CLK_chiplink_tx_clk	0	0
CLK_clk_hs_peri	0	0
CLK_div2_core	0	0
CLK_div2_hs_peri	0	0
CLK_div3_hs_peri	0	0
CLK_div4_core	0	0
CLK_div4_hs_peri	0	0
CLK_div4_peri	0	0
CLK_sdram_clk_o	0	0
CLK_spi_clk	0	0
CLK_spi_clk_out	0	0
CLK_u0_chiplink_rx_clk_pad_PAD	0	0
CLK_u0_clk_XC	0	0
CLK_u0_pll_FOUTPOSTDIV	0	0
CLK_u1_clk_XC	0	0

Fig. 5: iTO hold time reports without/with hold optimization.

timing path analysis, including setup and hold analysis, removal and recovery analysis, latch analysis, and multicycle analysis. Moreover, iSTA offers incremental timing propagation, detailed timing reports, advanced delay calculation, etc.

Functions. iSTA supports industry-standard format input data including verilog (netlist), sdc, spef, sdf and liberty files. If the input netlist is a hierarchy design, iSTA will flatten the netlist and analyze the timing performance. The delay calculation methods of iSTA can be divided into cell delay and connected net delay. The iSTA supports extensive multi-threads on read input data such as the timing-consume lib data, timing propagation including slew and delay propagation, and arrive time and require time propagation, etc. To achieve more authentic analysis, the iSTA tool supports basic on-chip variation (OCV) derate, and advanced OCV (AOCV). To support the low-power design, the iSTA supports clock-gate analysis. As the crosstalk effect becomes more important, the iSTA supports the basic crosstalk analysis. The delay deviations between iSTA and advanced commercial tools for all timing paths are considered as the quality metric of iSTA. After AI-based calibration, iSTA achieves impressive accuracy. A deviation and ratio comparison between iSTA and PT on open-source Skywater 130 process as shown in Fig. 4.

Implemented Key Technologies. Performance and analysis accuracy are two critical evaluations for timing analysis. The implemented cell delay includes the non-linear delay model (NLDM) and composite current source (CCS) model. The implemented connected net delay includes first-order Elmore, effective capacitance metric (ECM), second-order delay with two moments (D2M) model, modified D2M (MD2M), and high-order model with Arnoldi reduction. The iSTA uses a thread pool to manage the thread creation and destruction, which would make full use of CPU cores. To improve the calculation accuracy of cell delay, iSTA uses a machine learning method to achieve non-linear interpolation based on CCS model. To reduce the path delay deviation between iSTA and advanced commercial tools, we train a neural network to fit the two results and correct the deviation [15].

G. Timing Optimization

Timing optimization aims to ensure that the chip design is functionally correct and meets the design requirements for performance. iTO offers distinct and user-friendly interfaces to optimize timing design rule violations (DRVs), hold time violations, and setup time violations. Our goal is to eliminate all timing violations and enhance overall performance of chip.

Functions. The primary functions of iTO are to optimize timing DRVs, hold time violations, and setup time violations. iTO interacts with the iSTA to extract timing information from nets and timing paths. When violations are detected, iTO will insert buffer or adjust gate sizes to meet the timing constraints. Additionally, the built-in place legalization module automatically identifies the best legal location near the desired buffer location. iTO interacts with iSTA to generate a timing optimization report, which includes the pre- and post-optimization timing information, as well as details of the number of inserted buffers and gate sizing during the optimization process. An example report is shown in Fig. 5.

TABLE I: Comparison on power analysis results

cases	iPA total power	Innovus total power	deviation
aes_cipher_top	22.22mW	23.74mW	6.4%
gcd	0.38mW	0.37mW	3.6%
uart	0.51mW	0.49mW	3.9%

Implemented Key Technologies. iTO utilizes the HV-Tree or the rectilinear Steiner minimal tree (RSMT) constructed through FLUTE [16] to achieve the RC-tree of a net required by iSTA. To optimize timing DRVs, iTO employs a bottom-up traversal of the RSMT for the net. During this traversal, buffers are inserted to meet timing constraints, such as maximum load capacitance constraints. To optimize hold time, iTO first evaluates the potential delay introduced by delay buffers. It then inserts an appropriate number of delay buffers along the violating timing path based on the available timing slack. For setup time optimization, iTO implements a buffer insertion algorithm based on dynamic programming. This algorithm efficiently outputs the maximum time slack solution in quadratic time. In addition, to obtain a better setup time slack, iTO simultaneously optimizes buffer insertion and gate sizing by training a reinforcement learning (RL) model.

H. Power Analysis

iPA is a power analysis tool that supports both basic vectorless analysis without waveform data and vector analysis with VCD and SAIF data. It features fast toggle and static probability propagation.

Functions. iPA can read relevant data from iSTA and waveform data from VCD files, and generate comprehensive power analysis reports. iPA offers a range of essential power calculation algorithms, such as power graph construction, toggle propagation, static probability propagation, and static and dynamic power calculation for different types. To evaluate the accuracy of iPA on power analysis, we output the power consumption report and compare the results with the commercial tool Innovus on several test cases. TABLE I lists the comparison results, it can be seen that the average deviation is near 5%.

Implemented Key Technologies. To improve performance, iPA implements waveform compression and parallel techniques for reading VCD file. It proposes a depth-first search (DFS) technique based on latch thread pool for power graph construction and adopts thread pool technology for toggle propagation and static probability propagation.

I. Design Rule Check

iDRC is a design rule check (DRC) tool renowned for its dual capabilities in both local and global rule verification. It distin-

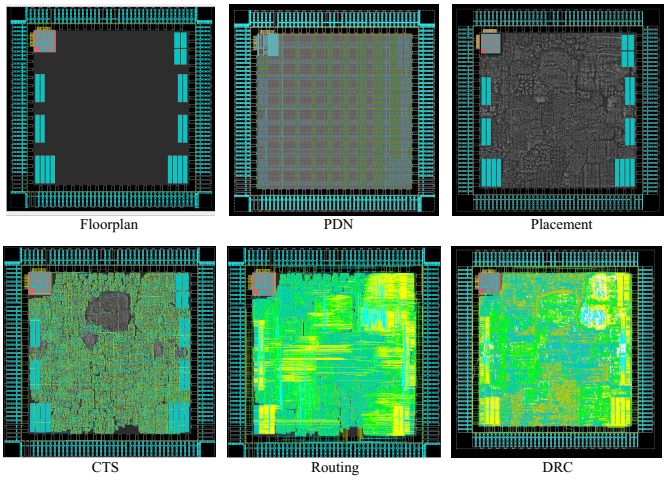


Fig. 6: The layouts from floorplan to routing designed by iPD.

guishes itself with its robust extensibility and comprehensive checking capabilities.

Functions. iDRC supports twenty-eight nano-meter and above process node geometric rule check of mainstream foundries. The fundamental back-end process design rules are mainly composed of metal layers and cut layers. The detailed geometric design rules include cut different layer spacing, cut end-of-line spacing, cut enclosure, cut enclosure edge, cut spacing, metal corner, filling spacing, metal end-of-line spacing, metal jog-to-jog spacing, metal notch spacing, metal parallel-run-length spacing, metal short, min hole, min step, minimal area. iDRC supports generating summary reports of design rules, allowing users to quickly assess the overall compliance of the layout. In addition, detailed reports are generated to pinpoint specific DRC violations within the layout.

Implemented Key Technologies. To achieve an efficient DRC tool, we have utilized advanced technologies like R-Tree data structures for region-based searching. This technique allows for efficient spatial indexing, which in turn reduces the computational complexity of rule checks. Moreover, we employ geometric computing algorithms to detect geometric shapes accurately.

J. Flow

To verify the functionality and effectiveness of iPD, we developed iFlow, which is implemented in Python by calling TCL-based or Python-based tool commands. It is worth noting that, since iPD supports only from netlist to GDS-II, iFlow integrates some commercial tools for signoff analysis and verification. We implement and tape out three chips with iPD: one 5-stage RISC-V chip supporting RT-Thread on 110nm; two 11-stage RISC-V chips supporting Linux on 110nm and 28nm in respective. An example of RISC-V chip design on 28nm process is designed by iFlow with iPD toolchain from Netlist-to-GDS II. The result and report of each step tool are shown in Fig. 6 and TABLE II.

IV. Conclusions

To provide an R&D platform on EDA designs, algorithms and tools, we present an open-source physical design EDA tool

TABLE II: Metrics on physical design steps.

part metrics	iPL (place)	iCTS	iTO	iRT (route)
#inst	1043440	1057291	1057549	1057549
#net	1015532	1029383	1029641	1029641
utilization	0.563929	0.570644	0.570768	0.570768
HPWL	34108823398	35042653984	35044866877	50157263995*
STWL	46195026227	46580611921	46581568292	
frequency	245.245	238.226	241.386	224.254
#DRC	0	0	0	233335

* Total wirelength after routing

design framework and a set of tools. To ensure maximum algorithm support, we have developed an EDA software architecture that emphasizes expandability through a decomposition and integration approach. Throughout the paper, we have introduced each tool in our iPD toolchain, highlighting their main functions, implemented key technologies, and evaluation metrics. Additionally, we have demonstrated a design flow with results and reports using the iPD toolchain. This toolchain provides a valuable resource for researchers and practitioners in the field of physical design.

Acknowledgment

This work is supported in part by the Major Key Project of PCL (No. PCL2023A03), the National Key R&D Program of China (No. 2022YFB4500403).

References

[1] M. S. Lundstrom and M. A. Alam, "Moore's law: The journey ahead," Science, vol. 378, no. 6621, pp. 722–723, 2022.
[2] T. Ajayi, D. Blaauw et al., "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain," in Proc. of GMACTC, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:210937106
[3] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," in Proc. of DAC, 2019.
[4] L. Liu, B. Fu, M. D. F. Wong, and E. F. Y. Young, "Xplace: An extremely fast and extensible global placement framework," in Proc. of DAC, 2022.
[5] H. Li, W.-K. Chow, G. Chen, B. Yu, and E. F. Young, "Pin-accessible legalization for mixed-cell-height circuits," IEEE Trans. on CAD, vol. 41, no. 1, pp. 143–154, 2022.
[6] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis, "Rsyn: An extensible physical synthesis framework," in Proc. of ISPD, 2017.
[7] R. T. Edwards, "graywolf," 2019. [Online]. Available: https://github.com/rubund/graywolf.
[8] ——, "Qrouter," 2019. [Online]. Available: https://github.com/RTimothyEdwards/qrouter/.
[9] J. Liu, C.-W. Pui, F. Wang, and E. F. Young, "CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model," in Proc. of DAC, 2020.
[10] H. Li, G. Chen, B. Jiang, J. Chen, and E. F. Young, "Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction," in Proc. of ICCAD, 2019.
[11] T.-W. Huang and M. D. Wong, "OpenTimer: A high-performance timing analysis tool," in Proc. of ICCAD, 2015.
[12] F. Huang, D. Liu, X. Li, B. Yu, and W. Zhu, "Handling orientation and aspect ratio of modules in electrostatics-based large scale fixed-outline floorplanning," in Proc. of ICCAD, 2023.
[13] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing," ACM Trans. on DAES, vol. 3, no. 3, pp. 341–388, Jul. 1998.
[14] G. Chen and E. F. Y. Young, "SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm," IEEE Trans. on CAD, vol. 39, no. 6, pp. 1217–1230, Jun. 2020.
[15] H. Liu, S. Wu, S. Tao, B. Xie, X. Li, and G. Li, "Accurate timing path delay learning using feature enhancer with effective capacitance," in Proc. of ISEDA, 2023.
[16] C. Chu and Y. C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," IEEE Trans. on CAD, vol. 27, no. 1, pp. 70–83, 2008.