Exploration of Open-source Physical Design Implementation Using OpenROAD

Darsi Madhavi
Junior Physical Design Engineer.
Rakiya Information Technology Solution ,
Bangalore, India
darsimadhavi.vlsi@gmail.com

Abstract— **In this paper, we discuss OpenROAD, a 7nm physical design flow that aims to develop open-source tools for "no human in the loop" digital systems that operate continuously generation of integrated circuit, packaging, and circuit boards. If OpenROAD is successful, it will assist achieve the IDEA goal of "democratisation of hardware design" by lowering the hurdles that system designers currently face related to cost, expertise, time, and risk. The 24-hour, no-humans goals of the IDEA programme are directly followed by a number of unique Technology directions. These include: Making machine learning prevalent in and around design tools and flows, simultaneous search and optimization to take advantage of the cloud resources available, dividing the problem and breaking it down to lower the latency of the solution, and layout generating techniques that offer "freedoms from choice" without significantly sacrificing design quality. However, creating open-source, self-driving design tools is a "moon shot" in and of itself, posing both technological and cultural difficulties. Timing error is now getting greater attention as a result of recent technical developments and the rising frequency of mistakes occurring on semiconductors. On the other hand, current methods for fixing timing issues are mostly concerned with time-delaying techniques and unnecessarily complicated procedures, which causes a timing issue on clock-based systems as well as hardware overhead . The proposed system of Clock Control Based Timing Error Tolerant System that uses a straightforward strategy to instantly correct a timing problem by changing the clock technique . The suggested approach improves throughput and increases the timing error tolerance of design.**

Keywords— OpenROAD Flow Scripts, RTL-to-GDSII flow, open-source tools, automated design, no-human-in-the-loop.

I. INTRODUCTION

The semiconductor industry has struggled to keep product design costs under control, even as hardware design tools and processes have evolved over the years. The implementation of hardware in advanced technologies is now hindered by hurdles related to cost, expertise, and unpredictability (risk). In other words, there are huge cost and risk hurdles to even starting hardware design, in addition to a local minimum of (i) complex and expensive tools, (ii) a lack of skilled users capable of using these tools in advanced technologies, and (iii) huge barriers to entry in terms of cost and risk for hardware design.[1-5]

Specifically in the field of digital integrated circuits (IC), In high-tech nodes, layout automation has been crucial to the creation of enormous, exceedingly complicated products. Even the most technologically advanced organisations, however, have been experiencing a lack of design capabilities for more than a decade. This shortcoming is the inability to expand product quality along with the growth of underlying device and patterning technologies [6] . Therefore, cutting-edge system-on-chip (SoC) product companies today must utilise specialisation and divide-and-conquer across large teams of designers in order to meet product and schedule requirements: each individual block of the design is handled by a separate subteam, and each designer has expertise in a specific facet of the design flow. DoD researchers and development teams typically experience hardware design cycles spanning 12 to 36 months since they lack the resources to carry out such a strategy.[7]

II. IDEA & THE OpenROAD FLOW

The 7nm physical design VSD Tapeout programme seeks to create a fully automated "no human in the loop" circuit layout generator that enables users without electronic design expertise to complete physical design of electronic hardware in order to get around the aforementioned restrictions and keep up with Moore's Law's exponential increases in SoC complexity. The OpenROAD ("Foundations and Realization of Open, Accessible Design") project [17] was launched.The primary objective of OpenROAD is to lower the financial, technical, and unpredictable hurdles that currently prevent system developers from accessing hardware implementation in cutting-edge technologies. OpenROAD aims to create a fully autonomous, open-source tool chain for digital layout generation across die, package, and board with an initial focus on the RTL-to-GDSII stage of system-on-chip design. The project's performers include Qualcomm, Arm, and a number of universities, including UC San Diego. To seed a future "Linux of EDA" (i.e., electronic design automation), we explicitly want to supply tapeout-capable tools in source code form with permissive licencing.[8]

The OpenROAD team's strategy to achieve no-human-in-loop (NHIL), 24-hour turnaround time is based on three cutting-edge basic technologies (TAT). First, the tool auto-tuning and design-adaptivity needed for NHIL will be made possible by machine learning based modelling and prediction of tool and flow outcomes.[9] Moreover, novel optimization cost functions in EDA tools that may expose to users. Secondly, extreme decomposition partitioning algorithms will allow thousands of tool copies running on cloud resources to optimise success while staying within human, CPU, and schedule limitations. With better optimizations and enhanced flow step predictability, quality loss from decomposition is recovered. Third, to maximise design outputs while staying within resource constraints and in the face of noise and chaos in the behaviour of complex , parallel/distributed search and optimization will make use of available compute resources (such as the cloud). Using "freedoms from choice" in layout generation to simplify design and tool complexity can boost predictability and reduce the need for design revisions.

III. A NEW APPROACH

The contributions and strategy of OpenROAD aim to develop a new paradigm for EDA tools, academic-industry collaboration and academic research itself. OpenROAD intends to ultimately overcome established, "cultural," and "critical mass/critical quality" impediments to developing an open-source mindset in the EDA industry. To begin the project, we bring (i) in a commercial static timing analysis tool and considerable initial software IP, including donated source code bases; (ii) a substantial collection of academic software IP and skill sets;(iii) top-tier expertise in SoCs and IP and direction from partners in the industry Qualcomm and Arm;(iv) a built-in internal design team (from the University of Michigan) to offer de facto product engineering and beta customer services; and (v) a comprehensive plan for industry and academic outreach. Moreover, OpenROAD draws directly from the IDEA programme requirements for its "Base Technology" activities (no-humans, 24-hours, no loss of PPA quality). In our opinion, achieving the IDEA aim requires the cohesive integration of machine learning, issue division and decomposition, and parallel/distributed search and optimization.

This article :The tools and flow for OpenROAD that are currently deployed on GitHub will be described in more detail later on in this article. Early proof points and calibrations for the "RTL-to-GDSII" method of creating digital IC layouts, which includes 7nm technology, have been attained.

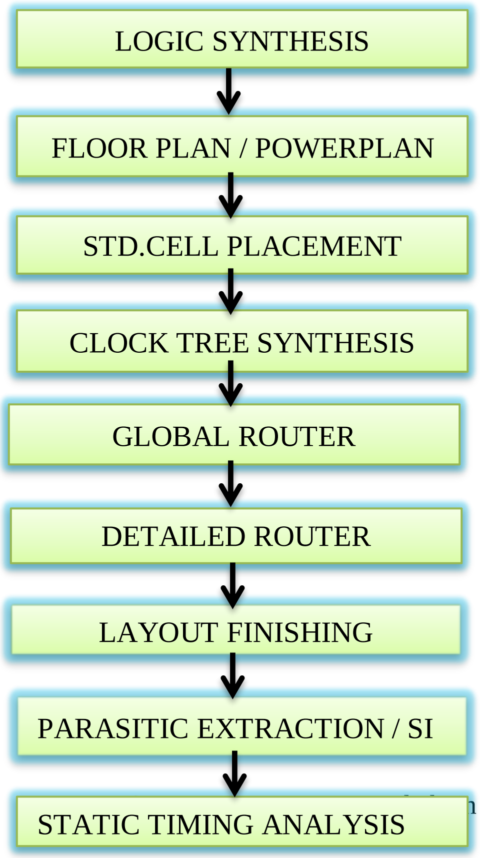

Figure 1 : OPEN ROAD FLOW

The layout generation tool chain for OpenROAD comprises of a collection of open-source programmes that attempt to produce tapeout-ready GDSII files from RTL Verilog, constraints (.sdc), liberty (.lib), and technology (.lef) files. The flow of tools related to specific OpenROAD tasks is shown in Figure 1. They include Logic Synthesis (LS), Floorplan (FP) and Power Delivery Network (PDN), Placement , Clock Tree Synthesis (CTS) and Routing

A. Logic Synthesis (LS)

Timing awareness and optimization are a significant open-source LS gap. Timing-driven synthesis can be enabled in two ways, according to OpenROAD. To begin with, we employ machine learning strategies to enable autonomous design space explorations for timing-driven logic improvement.

Synthesis scripts frequently contain tens of commands to enable a design to achieve its temporal and area objectives. These scripts were written by human experts . We create machine learning agents that automatically construct step-by-step synthesis scripts to achieve target timing and delay goals in order to produce the best synthesis scripts that are suited to specific circuits.

The RePlAce [20] placement tool is integrated into the logic synthesis cycle to provide physical-aware logic synthesis, which uses global placement-based wire capacitance estimates to enhance timing outcomes.Our ultimate goal is to feed back wire estimations as they are refined in physical design processes (such as standard-cell placement and global routing) to improve synthesis findings because current academic tools are blind to the results of following steps in the design flow.

B. Floorplan and PDN

TritonFPlan, which includes two main parts, performs floorplanning and power delivery network synthesis. The first part is a mixed-size integer programming-based macro block packing that takes macro-level connection into account (blocks and standard cells) global positioning.The second element is the creation of a power delivery network (PDN) using Tcl and a safe-by-construction methodology. In order to use TritonFPlan, the user must supply a number of configuration files, such as IP global.cfg and IP local.cfg, which collect macro packing rules, and PDN.cfg, which captures information on safe-by-construction metal and through geometry. The difficulty of academic open-source tool developers (or, their tools) to see all unencrypted design enablements from the foundry has led to the need for these configuration files.In order to initialise its global placement, the TritonFPlan tool uses the mixed-size placer (RePlAce). A starting point from which various floorplan solutions can be developed is the generated macro global locations. We once again use our placer (RePlAce) to select the optimum floorplan based on an estimated total wirelength criteria for each of the generated floorplan options with fixed macros and PDN. Only supporting rectangular floorplans and macro counts under 100 are some of the restrictions.

C. Placement

A BSD-licensed open-source analytical placer based on the electrostatics analogy is called RePlAce [10-13]. RePlAce is used in OpenROAD for mixed-size (macros and cells) placement during floorplanning, for standard-cell placement inside a certain floorplan, and for clock buffer legalisation during clock tree synthesis (CTS) [14]. Timing-driven placement is accomplished by the combination of FLUTE [15] and OpenSTA [16], as well as an iterative signal net reweighting [17]. Standard LEF/DEF, Verilog, SDC, and Liberty input formats are accepted by the timing-driven TDRePlAce tool, which also has a quick RC estimator for parasitics extraction. The goal of ongoing efforts is to make commercial format (LEF/DEF/Verilog) available in routability-driven mode.The University of Michigan internal design advisors subteam's creation, a tiny RISCV-based block (foundry 16nm technology), is placed in RePlAce.

D. Clock Tree Synthesis

Based on the GH-Tree (generalised H-Tree) paradigm of [18], TritonCTS [19] performs clock tree synthesis (CTS) for low-power, low-skew, and low-latency clock distribution. A clock tree topology that has the least predicted power and is compatible with the provided latency and skew targets is found through a dynamic programming approach. Clock buffer placement and sink clustering are carried out using linear programming. Either the single-trunk Steiner tree or the Prim-Dijkstra [20] method can be used for leaf-level routing.TritonCTS features interfaces with the placer (RePlAce) and the router in the layout generation flow (TritonRoute [21]).

The placer is used to legalise clock buffer inserts.The sink pins that should be used for clock tree routing are mapped by the router to GCELLs. LEF, placed DEF, placed gate-level Verilog, a configuration file, and library characterization files are the inputs for TritonCTS. (Each foundry enablement requires once-only library characterization.Today, a third party (foundry or tool user) is anticipated to carry out this library characterisation using commercial EDA tools.).The TritonCTS outputs use "buffered" gate-level Verilog, "buffered" placed DEF, and "buffered" clock tree global routing [22]. On GitHub, TritonCTS is accessible to everyone [23]. Using 16nm and 28nm foundry enablements, preliminary validations have been made. Multiple clock sources handling, non-default routing rules handling, etc. are still being improved.

E. Routing

Using a global routing solution in route guide format [24], TritonRoute [25] consumes LEF and placed DEF before performing detailed routing for both signal nets and clock nets. TritonRoute uses a quick approximation approach to preprocess the global routing solution before the detailed routing to guarantee that each net has a Steiner tree structure. In the detailed routing step, congestion and wirelength are reduced while maintaining net connectivity. After then, the detailed routing problem is iteratively solved layer by layer, with each layer being divided into separate routing panels.
A mixed integer linear programming (MILP)-based method is used to solve the maximum weighted independent set (MWIS) version of the panel routing problem in parallel. The MWIS formulation distributes tracks in the best possible way while taking into account (i) wirelength and via minimization, (ii) intra- and inter-layer connection, and (iii) different design rules.

REFERENCES

[1] https://vlsicad.ucsd.edu/Publications/Conferences/395/c395.pdf 2022

[2] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[3] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[4] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds.New York: Academic, 1963, pp. 271–350.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[8] “IEEE CEDA Design Automation Technical Committee,” https://ieee-ceda.org/ node/2591.

[9] “IEEE CEDA DATC Robust Design Flow,” https://github.com/ieee-ceda-datc/ datc-robust-design-flow

[10] J. Jung, I. H.-R. Jiang, G.-J. Nam, V. N. Kravets, L. Behjat, and Y.-L. Li, “OpenDesign Flow Database: The infrastructure for VLSI design and design automation research,” Proc. ICCAD, Nov. 2016, pp. 42:1–42:6.

[11] J. Jung, P.-Y. Lee, Y. Wu, N. K. Darav, I. H. Jiang, V. N. Kravets, L. Behjat, Y. Li, and G. Nam, “DATC RDF: Robust design flow database,” Proc. ICCAD, Nov. 2017, pp. 872–873.

[12] J. Jung, I. H.-R. Jiang, J. Chen, S.-T. Lin, Y.-L. Li, V. N. Kravets, and G.-J. Nam, “DATC RDF: An academic flow from logic synthesis to detailed routing,” Proc. ICCAD, Nov. 2018, pp. 37:1–37:4.

[13] —, “DATC RDF: An open design flow from logic synthesis to detailed routing,” Proc. Workshop on Open-Source EDA Technology (WOSET), Nov. 2018, pp. 6:1–6:4.

[14] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, “DATC RDF-2019: Towards a complete academic reference design flow,” Proc. ICCAD, Nov. 2019, pp. 1–6.

[15] T. Ajayi, V. A. Chhabria, M. Fogaca, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, “Toward an open-source digital flow: First learnings from the OpenROAD project,” Proc. DAC, June 2019, pp. 76:1–76:4.

[16] “The OpenROAD Project,” https://github.com/The-OpenROAD-Project

[17] KLayout,” https://www.klayout.de/.

[18] “OpenAccess Silicon Integration Initiative,” https://si2.org/openaccess/.

[19] “OpenLANE,” https://github.com/efabless/openlane.

[20] “OpenCores,” https://opencores.org/.

[21] “OpenSTA,” https://github.com/The-OpenROAD-Project/OpenSTA.

[22] “Routing With Cell Movement,” ICCAD-2020 CAD Contest, Problem B. http:// iccad-contest.org/2020/problems.html.

[23] “DATC RDF Timer Calibration,” https://github.com/ieee-ceda-datc/ datc-rdf-timer-calibration.

[24] Developing Industrial Strength EDA Tools Using the OpenDB Opensource Database and the OpenROAD Framework,” https://github.com/ The-OpenROAD-Project/DAC-2020-Tutorial.

[25] A. B. Kahng, J. Li, and L. Wang, “Improved flop tray-based design implementation for power reduction,” Proc. ICCAD, Nov. 2018, pp. 1–8.

[26] Y. Chang, T.-W. Lin, I. H.-R. Jiang, and G. Nam, “Graceful register clustering by effective mean shift algorithm for power and timing balancing,” Proc. ISPD, Apr, 2019, pp. 11–18

[27] A. B. Kahng, “Reducing time and effort in IC implementation: A roadmap of challenges and solutions,” Proc. DAC, June 2018, pp. 36:1–36:6.

[28] D. Aldous and U. Vazirani, ““Go with the winners” algorithms,” Proc. IEEE Symp. on Foundations of Computer Science, 1994, pp. 492–501.

[29] A. B. Kahng, I. Kang and S. Nath, “Incremental multiple-scan chain ordering for ECO flip-flop insertion,” Proc. ICCAD, Nov. 2013, pp. 705–712.

[30] “Fault,” https://github.com/Cloud-V/Fault.

[31] “SkyWater Open Source PDK,” https://github.com/google/skywater-pdk

[32] P. Chakraborty, J. Cruz, and S. Bhunia, “SAIL: Machine learning guided structural analysis attack on hardware obfuscation” in Proc. AsianHOST, Dec. 2018, pp. 56–61.

[33] C. Wolf and J. Glaser, “Yosys-a free Verilog synthesis suite,” in Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip), 2013.

[34] “Open Circuit Design,” http://opencircuitdesign.com.

[35] T. Spyrou, “OpenDB, OpenROAD’s Database,” in Proceedings of the Workshop on Open-Source EDA Technology (WOSET), 2019.