Development of an Interactive Graphical User Interface for ASIC Floorplanning

IE0499 - Electrical Project

Second Semester 2025

Elizabeth Matamoros Bojorge
University of Costa Rica
School of Electrical Engineering
elizabeth.matamaros@ucr.ac.cr

***Abstract*—The physical design of integrated circuits includes the floorplanning stage, in which the chip dimensions, the core region, and the global organization of blocks and pins are defined. This work presents the development of an interactive tool to support this stage within an open-source ASIC design flow. The main objective is to provide a graphical interface that allows users to visualize a synthesized design, manually rearrange its logic blocks, and export floorplan configurations compatible with the OpenLane flow. The methodology consists of taking netlists generated by open-source synthesis tools, building from them an intermediate model of the design (modules and interconnections), and developing a Python application that represents each block as a movable rectangle on a canvas. The tool computes simple metrics, such as an approximate half-perimeter wirelength (HPWL) and pin counts, and generates configuration files that can be fed back into OpenLane to run the complete physical flow. The results obtained with a test design show that the interface enables intuitive exploration of different floorplan alternatives and the production of valid configurations for physical implementation. It is concluded that the tool facilitates early analysis of spatial organization decisions and is useful in academic contexts, while leaving room for the future incorporation of more advanced placement algorithms and additional evaluation metrics.**

I. INTRODUCTION

The physical design of digital integrated circuits is the stage of the VLSI design flow in which a logical netlist is transformed into a concrete geometric layout, ready for fabrication. In this phase, the distribution of logic blocks, standard cells, memories, and power structures within the chip area is determined, and physical routes for interconnections are defined while meeting timing, area, and power constraints. According to Kahng et al. [1], chip planning encompasses floorplanning, pin assignment, and power planning, and floorplanning in particular determines the locations and dimensions of the blocks resulting from circuit partitioning, enabling early estimates of interconnect length, delay, and overall chip performance.

In industrial practice, floorplanning is usually regarded as the first major decision within physical design, as it establishes the size and shape of the core, reserves space for power and signal routing, and fixes the location of macros and standard-cell placement regions. As physical-design flow guidelines for digital design point out, the quality of the floorplan directly affects the difficulty of later stages such as placement, routing, and timing closure [2]. A poor floorplan can lead to excessive wirelength, severe congestion, or even the impossibility of achieving timing closure, even when advanced placement and routing algorithms are available.

At the same time, floorplanning is one of the stages in which the designer retains the greatest room for manual intervention. Most modern flows heavily automate logic synthesis, standard-cell placement, and routing, whereas decisions about the global organization of the chip, the relative position of functional blocks, and the distribution of pins still depend largely on the intuition and experience of the engineering staff. This is particularly evident in open-source tool environments: flows such as OpenLane provide a complete implementation chain from RTL to GDSII, integrating synthesis, placement, routing, and physical verification within an automated environment [3], but offer limited interactive support for exploring, comparing, and refining floorplan alternatives before launching the full physical flow.

From an academic perspective, this situation opens a clear area of interest. On the one hand, physical design and floorplanning are discussed in specialized texts and advanced courses [1], but learning often relies on commercial tools that are difficult for students to access. On the other hand, open flows based on OpenLane allow a complete physical implementation to be executed, but the decision-making process around the floorplan tends to remain hidden behind configuration files and scripts. Having interactive tools integrated with these flows, which allow users to visualize the structure of the design, manipulate blocks, and observe the effect of those decisions on simple interconnection metrics, is valuable both for education in VLSI design and for exploring alternatives in real projects.

In this context, the present work is framed, whose overall objective is to develop an interactive graphical interface for floorplanning exploration within an open ASIC physical design flow. The proposal is to use netlists generated by tools such as Yosys and integrated into OpenLane-type physical implementation flows [3] as a starting point, build from them an intermediate model of the design (modules, instances, and connectivity), and provide the user with a visual representation in which each block is shown as a movable rectangle on a canvas. In this way, the designer can manually rearrange the blocks, adjust their distribution within the chip area, and propose alternative floorplan configurations in a more intuitive manner.

The methodology combines software development in Python with the use of graphical libraries and utilities associated with the open design flow. In a first stage, a test design is defined and a synthesized netlist compatible with OpenLane is obtained. From this netlist, an intermediate file is generated that describes the modules, their hierarchy, and the connections between them. In a second stage, a GUI is built that allows these modules to be visualized, dragged, and have their relative positions modified and, eventually, associated with regions of the core area and edges reserved for pins. On top of this representation, simple metrics are computed, such as an approximate interconnect length using the half-perimeter wirelength (HPWL) of selected nets, the number of pins involved in critical connections, and qualitative observations about potential congestion hot spots.

The ultimate goal of this development is for the floorplan configurations generated in the graphical interface to be exported back into the implementation flow, for example through configuration files compatible with OpenLane (area parameters, macro positions, pin ordering), thus enabling a complete experimentation loop: proposing a floorplan in the GUI, exporting it, running the physical flow, and evaluating whether the decisions made are reflected in improved routing, timing, or area utilization results. In this way, the work does not aim to replace existing placement algorithms, but rather to provide an intermediate layer of inspection and manual control over a particularly sensitive stage of physical design.

For reasons of scope and time, the project focuses on building the basic infrastructure for this integration: netlist ingestion, graphical representation of the design, computation of simple metrics (such as HPWL and pin counts), and generation of exportable configurations. The incorporation of automatic placement algorithms within the same graphical environment, the systematic evaluation of multiple test designs, and the integration of more advanced congestion and timing metrics are identified as future work. The remainder of the document details the methodology and tools used, presents the results obtained for the chosen test design, and discusses the main conclusions and possible extensions of this development.

II. METHODOLOGY

The development of the project was organized according to the specific objectives defined for the floorplanning support tool. The adopted methodology was iterative: it began with the selection of technologies and the construction of an initial prototype for netlist processing, then progressed with the design and validation of the interactive graphical interface, and finally addressed the integration with the OpenLane flow through export mechanisms and tests on designs of varying complexity.

In the first phase, the programming language and core libraries were selected. Python was chosen as the main language, given its rich ecosystem of libraries for file handling, structured formats, and graphical user interface development, as well as its good integration with open-source design tools. Based on this decision, the script ingest_netlist.py was developed, whose purpose is to automatically process the structural information of the design. This script takes as input a netlist generated by Yosys from RTL code and transforms it into an intermediate JSON file suitable to be consumed by the GUI. During this stage, the logic required to extract only the information relevant to floorplanning was implemented: the list of instantiated modules, the connections between them (nets), input and output ports, and basic parameters associated with the chip area. The original netlist is filtered so that information unnecessary for visualization is removed, yielding a compact representation focused on the hierarchical structure and interconnectivity of the design.

The second phase focused on the design and development of the interactive graphical interface, corresponding to the specific objectives related to visual floorplan exploration. Based on the JSON file generated by the ingestion script, a graphical representation was built in which each module is shown as a rectangular block, with dimensions proportional to the available area on the die. The interface implements the ability to freely move these blocks across a canvas, subject only to a non-overlap constraint and preserving the geometric limits of the chip. A mechanism was also designed to place and move pins exclusively along the die perimeter, preventing them from being positioned in invalid regions. In addition, visual aids were incorporated to facilitate floorplan analysis: module and pin counts, approximate half-perimeter wirelength (HPWL) computation, and a wiring representation using thickness and color encoding. In particular, a scale was defined in which single-bit connections are drawn as thin, soft-colored lines, whereas nets with higher bit density are represented with thicker lines and more intense colors, such as reds to indicate potentially congested regions. This logic is updated in real time so that any movement of blocks or pins is immediately reflected in both the metrics and the visualization.

In a third phase, the integration of the tool with the OpenLane physical design flow was addressed, fulfilling the specific objective related to configuration export. To this end, an export mechanism was implemented that, based on the current floorplan state in the GUI, generates the files required by the flow: a macro.cfg file for the placement of blocks or macros, a pin_placement.cfgfile for the location of pins along the chip perimeter, and a config.tcl file containing the general design configuration. A particularly important aspect of this stage was ensuring that the pin description was expan ded bit by bit, in accordance with OpenLane's expectations. Initially, the tool described buses in aggregated form, which prevented the flow from running correctly. This issue was resolved by implementing bus expansion in the export phase: the interface internally preserves the notion of each connection's width (needed for the metrics and visual encoding), but the output file decomposes each bus into individual pins, compatible with the syntax required by the physical flow.

Throughout the procedure, multiple test cycles were carried out on designs of varying complexity to validate netlist ingestion, the GUI, and the export mechanisms. Initially, small RTL designs with few modules and connections were used to verify the correct mapping of instances and pins, as well as the basic response of the interface when moving blocks and regenerating metrics. Later, OpenLane reference examples were employed, such as a design based on PicoRV32, and test netlists were generated with different bus widths and numbers of modules, gradually scaling from 5 and 7 modules up to cases with around 30 modules represented simultaneously in the interface. These tests made it possible to detect issues such as pins escaping the die perimeter, inconsistencies between the graphical representation and the netlist signal names, and configuration errors in the config.tcl file. Each of these issues was solved incrementally by refining the geometric constraints in the GUI, adjusting the logic for pin name generation, and debugging configuration parameters until complete OpenLane flow executions were achieved without errors. In this way, the procedure followed was not linear but iterative, combining design, implementation, and continuous testing until converging on a functional tool integrated with the physical design flow.

III. RESULTS

The developed tool was evaluated through a series of progressive tests on RTL designs of varying complexity, with the goal of verifying the correct operation of each component: netlist ingestion, interactive floorplan visualization, metric computation, and export of configuration files to OpenLane. In all cases, it was confirmed that the flow could be completed without errors, yielding implementations consistent with the configurations proposed in the GUI.

In an initial testing stage, small designs with few modules and narrow bus widths were used to validate the netlist processing step. Based on netlists generated by Yosys, it was confirmed that the ingestion script correctly produces the intermediate JSON file, preserving the list of instances, the connectivity between modules, and the names of input and output ports. These tests made it possible to detect and correct details such as the handling of simple hierarchies and the need to explicitly represent bus widths; once these issues were resolved, JSON generation became stable and repeatable across different inputs.

Subsequently, the behavior of the interactive graphical interface was evaluated using medium-sized designs with a larger number of signals. In these cases, the GUI displayed all expected modules as rectangular blocks within the die area, allowing their free movement and the relocation of pins along the chip perimeter. It was verified that the implemented metrics—module count, pin count, and approximate half-perimeter wirelength (HPWL)—are updated in real time in response to any change in the floorplan. Qualitatively, it was observed that bringing strongly connected modules closer together led the tool to report a reduction in HPWL and in the apparent length of the connections, whereas more dispersed configurations produced higher values, which is consistent with the expected behavior of this metric.

A relevant aspect of the results was the visual representation of wiring. The interface clearly distinguished connections of different widths by varying line thickness and color to indicate higher or lower bit density. This made it easier to quickly identify regions of the design with a high concentration of interconnections and motivated adjustments in the position of modules and pins to reduce potential congestion hot spots. In the tests conducted with designs reaching approximately 30 modules on screen, the interface remained usable, and the graphical updates continued to correctly reflect the changes made by the user.

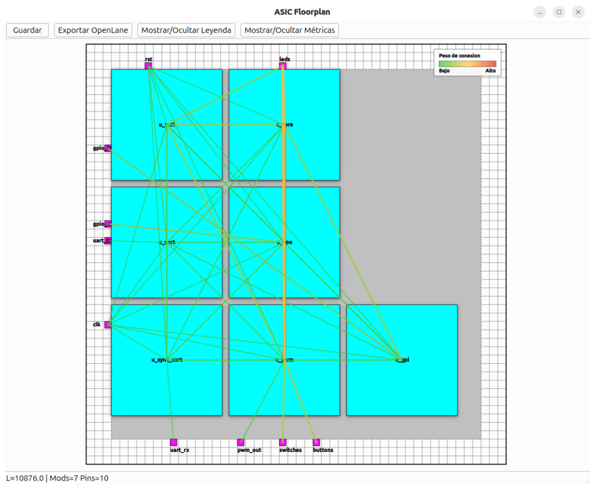
Regarding export to the OpenLane flow, it was verified that the tool consistently generates the macro.cfg, pin_placement.cfg, and config.tcl files based on the current floorplan state in the GUI. During the first runs, it was detected that the pin description grouped entire buses, which prevented OpenLane from accepting the configuration. After introducing bit-by-bit expansion in the export stage, the resulting files became compatible with the flow's requirements. Following this correction, all test designs used were successfully executed in OpenLane through the complete physical flow, yielding valid placement, routing, and GDS results.

To illustrate the functionality of the developed interface, a case-study experiment was carried out using one of the RTL designs employed in the tests. Starting from the same design, composed of 7 modules and 10 pins, five floorplan variants were generated by modifying only the relative positions of the modules and the locations of the pins along the die perimeter. For each variant, the tool computed the approximate half-perimeter wirelength (HPWL), enabling a direct comparison of how spatial organization decisions affect this metric.

The obtained results show HPWL values of 10876.0, 11956.0, 8698.0, 9032.0, and 8580.0 for configurations 1 through 5, respectively. Fig. 1 illustrates the initial layout (design 1), in which the modules are distributed more uniformly and several long connections crossing the chip can be observed, whereas Fig. 2 corresponds to floorplan 5, where the modules with the highest number of interconnections were regrouped in the lower central area. In this latter configuration, the HPWL is reduced to 8580.0, representing a significant improvement over the initial value. The line thickness and color encoding helps identify the most critical connections and, overall, this example shows how the interface enables the exploration of different floorplanning alternatives, the visualization of their immediate impact on the metrics, and guidance toward configurations with better quality from the standpoint of interconnect length.

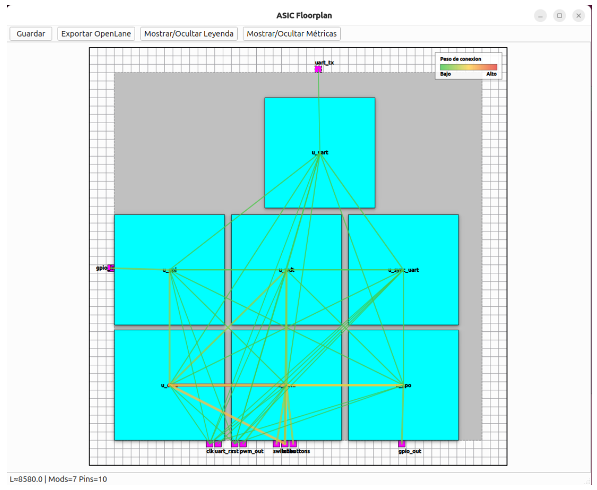
Table I
HPWL COMPARISON FOR FIVE FLOORPLAN VARIANTS

Floorplan	Approximate HPWL
1	10876.0
2	11956.0
3	8698.0
4	9032.0
5	8580.0



L≈10876.0 | Mods=7 Pins=10

Figure 1. Interface for floorplan 1 (HPWL = 10876.0)



L≈8580.0 | Mods=7 Pins=10

Figure 2. Interface for floorplan 5 (HPWL = 8580.0)

IV. CONCLUSIONS

The work carried out made it possible to achieve the proposed overall objective: to develop an interactive graphical interface that allows the floorplan of an ASIC to be visualized and manually modified from netlists generated within an OpenLane flow using the SKY130 PDK, thus facilitating the exploration of alternative configurations before the placement and routing stage. To this end, a workflow was consolidated that starts from the synthesized netlist, transforms it into an intermediate model (design.json), and presents it in a GUI where it is possible to rearrange blocks and pins, as well as export the resulting configurations back into the physical design flow.

One of the central contributions of the project was to show that it is possible to insert a layer of manual inspection and control over the floorplan without breaking the automated OpenLane flow. The development of the ingestion script made it possible to filter the relevant information from the netlist, capturing modules, hierarchy, and connectivity in a compact format, while the graphical interface provided a global view of the chip, with movable blocks and pins constrained to the die perimeter. The inclusion of metrics such as the approximate half-perimeter wirelength (HPWL) and pin counts, together with connection encoding by line thickness and color, proved sufficient to highlight clear differences between floorplanning alternatives and to guide adjustments in the position of modules and pins toward configurations with shorter interconnect length.

The experimental results obtained reinforce this idea. In particular, the case study on a design with seven modules and ten pins showed that changes in spatial organization can lead to significant variations in HPWL, from values close to 12000 down to configurations around 8500 units. Moreover, all the considered floorplan variants could be correctly exported to macro.cfg, pin_placement.cfg, and config.tcl files, and the OpenLane flow reached a valid physical implementation in every case. In this way, the project validates that integrating an interactive interface with an open physical design flow is feasible and provides concrete benefits in terms of understanding and exploring alternatives.

From an educational perspective, the development and use of this tool adds additional value. It becomes possible to immediately observe how block and pin placement decisions affect interconnection metrics, which helps build a stronger intuition about physical design, traditionally seen as a "black box" dominated by commercial tools. At the same time, in prototyping and exploration contexts, the GUI makes it possible to try out floorplanning configurations without the need to manually edit configuration files, reducing errors and speeding up the test-and-feedback cycle.

Nevertheless, the project also highlights some limitations and opportunities for improvement. The tool focuses on simple geometric metrics and manual block manipulation, without yet integrating more advanced models of timing, power consumption, or routing congestion. In addition, the tests were carried out on small and medium-sized designs; although the results are promising, it would be necessary to study in more detail the behavior of the interface when dealing with large-scale designs and deeper hierarchies. Moreover, the quality of the floorplan still depends largely on the user's judgment, which opens the door to incorporating automatic algorithms that suggest reasonable initial placements or adjustments guided by metrics.

Finally, as future work, the integration of placement algorithms that operate on the same data model used by the GUI is proposed, enabling the combination of interactive exploration and automated optimization. Likewise, it would be desirable to extend the set of available metrics (for example, simple congestion estimates or indicators of signal and power load balance), support multiple technologies and open design flows, and even evaluate the tool through user studies to measure its impact on the learning process. Taken together, the results obtained and the possible extensions show that this type of tool has significant potential to bring ASIC physical design closer to the academic environment.

References

[1] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, VLSI Physical Design: From Graph Partitioning to Timing Closure. Cham: Springer, 2nd ed., 2022.

[2] VLSI Pro, "Physical design flow i: Netlistin & floorplanning." https://vlsi.pro/physical-design-flow-i-netlistin-floorplanning/, 2015.

[3] A. A. Ghazy and M. Shalan, "OpenLANE: The open-source digital ASIC implementation flow," in Proc. Workshop on Open-Source EDA Technology (WOSET), 2020.