AMBEATion: Analog Mixed-Signal Back-End Design Automation with Machine Learning and Artificial Intelligence Techniques

Giulia Elena Aliffi*[2], Joao Baixinho[¶], Dalibor Barri[§], Francesco Daghero[‡], Nicola Di Carolo[§], Gabriele Faraone[§], Michelangelo Grosso[§], Daniele Jahier Pagliari[‡], Jiri Jakovenko[†], Vladimír Janíček[†], Dario Licastro[§], Vazgen Melikyan[¶], Matteo Risso[‡], Vittorio Romano*[1], Eugenio Serianni[§], Martin Šťastný[†], Patrik Vacula[§], Giorgia Vitanza*[2], Chen Xie[‡]

*University of Catania, Catania, Italy, E-mail: name.surname@unict.it[1]; name.surname@phd.unict.it[2]

[†]Czech Technical University, Prague, Czech Republic, E-mail: name.surname@fel.cvut.cz

[‡]Politecnico di Torino, Turin, Italy, E-mail: name.surname@polito.it

[§]STMicroelectronics, E-mail: name.surname@st.com

[¶] Synopsys, E-mail: name.surname@synopsys.com

***Abstract*—For the competitiveness of the European economy, automation techniques in the design of complex electronic systems are a prerequisite for winning the global chip challenge. Specifically, while the physical design of digital Integrated Circuits (ICs) can be largely automated, the physical design of Analog-Mixed-Signal (AMS) ICs built with an analog-on-top flow, where digital subsystems are instantiated as Intellectual Property (IP) modules, is still carried out predominantly by hand, with a time-consuming methodology. The AMBEATion consortium, including global semiconductor and design automation companies as well as leading universities, aims to address this challenge by combining classic Electronic Design Automation (EDA) algorithms with novel Artificial Intelligence and Machine Learning (ML) techniques. Specifically, the scientific and technical result expected at the end of the project will be a new methodology, implemented in a framework of scripts for AMS placement, internally making use of state-of-the-art AI/ML models, and fully integrated with Industrial design flows. With this methodology, the AMBEATion consortium aims to reduce the design turnaround-time and, consequently, the silicon development costs of complex AMS ICs.**

***Index Terms*—EDA, analog design, layout, analog Back-End, machine learning, artificial intelligence, analog-mixed-signal, integrated circuits**

I. Introduction

Improving the development process of electronic systems is an essential factor for the competitiveness of the European economy. In fact, most key technology areas at the forefront of tomorrow's society, including among others the Internet of Things (IoT), Smart Driving, Industry 4.0, and Active Assisted Living (AAL), rely on complex electronic devices. These Smart Systems, are composed of heterogeneous parts, providing different functionalities, including digital and analog blocks, sensors, actuators, power generation and storage, etc. In this context, on-chip heterogeneous integration is fundamental to improve perfomance (e.g., speed, reliability and power consumption) and cost-effectiveness. Therefore, a leading role is held by Analog Mixed-Signal Integrated Circuits (AMS-ICs) where Analog and Digital domains are strictly intertwined.

In AMS ICs, digital blocks usually carry out computational tasks, and their design and physical implementation are highly automated by means of highly specialized flows and optimized standard cell libraries [1]. On the other hand, analog parts realize functionalities such as power conversion, Analog-to-Digital Conversion (ADC), Digital-to-Analog Conversion (DAC), thermal sensing, voltage sensing, current sensing, actuation, signal conditioning elements, etc. Despite numerous efforts [2]–[4], automated and universally valid analog physical design flows are not yet available. The reason for this discrepancy is rooted in the inherently higher difficulty of the analog layout problem. In fact, not only are analog circuits more sensitive to noise and variability effects (Well Proximity Effect - WPE, Shallow Trench Isolation - STI stress, thermal gradient, mechanical stress, process variation - mismatch, etc.) but they also have larger and more complex sets of constraints (e.g., gain, bandwidth, distortion, etc), all of which affect performance [5], [6]. Additionally, different specific classes of analog circuits require different metrics and trade-offs [6].

Machine Learning (ML) and Artificial Intelligence (AI) techniques have been introduced in the Electronic Design Automation (EDA) industry since several years to improve the efficiency and scalability, as well as the Quality of Results (QoR) of design and verification tools. ML has been applied ubiquitously, from synthesis to floorplanning, place&route, timing analysis, analog design and simulation [7], [8]. However, while digital flows greatly benefits from the integration with ML and AI techniques [7], [8], a wide gap still exists in this respect within the analog domain [9].

This paper introduces the Marie Skłodowska-Curie Research and Innovation Staff Exchange project **AMBEAtion** (**A**nalog **M**ixed-signal **B**ack-**E**nd design **A**utoma**tion** with

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101007730.

979-8-3503-4859-0/DATE24/© 2024 EDAA

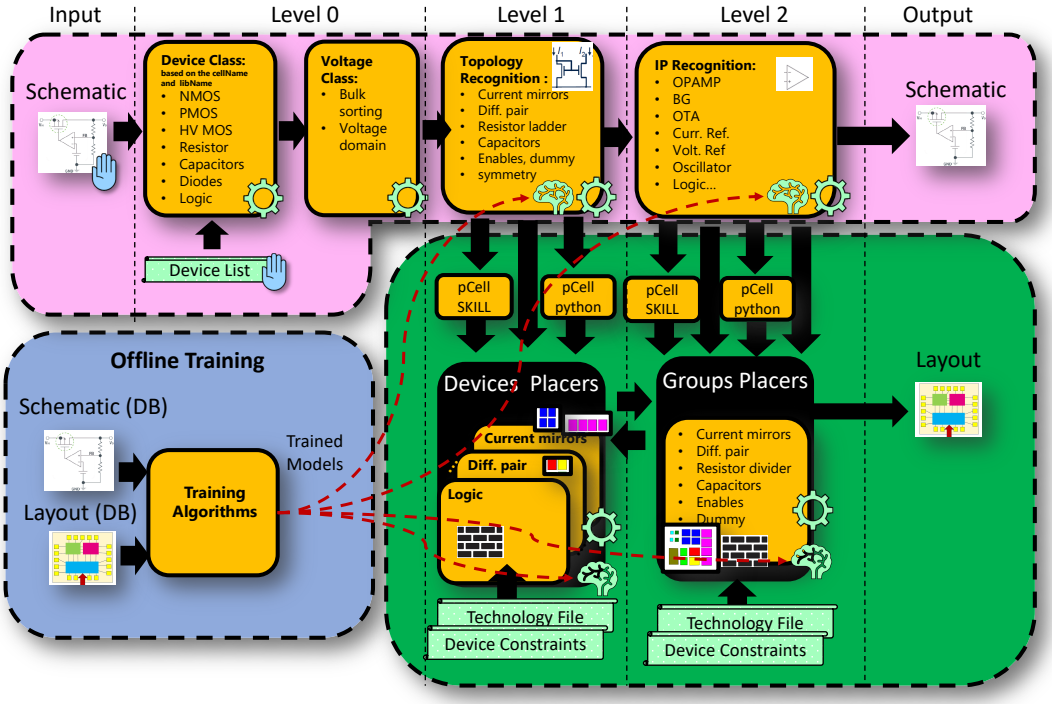

Fig. 1. The AMBEATion flow.

Machine Learning and Artificial Intelligence Techniques). The project aims to improve the quality and productivity of AMS physical design, with particular focus on device placement, by developing a new methodology implemented in a framework of scripts, code parsers, automated processes and simulation tools, internally making use of a combination of classic EDA algorithms, and novel AI and ML models. A further intent of the AMBEATion project is to reach full integration with Industrial Design flows by leading EDA tool providers, thus allowing a seamless addition of the methodology to existing AMS designers' workflows, as well as easing the access to large databases of past designs.

The scientific contributions within the AMBEATion consortium are, therefore, strongly driven by the needs of industrial partners, while academic partners contribute with their expertise on AI/ML. The presence in the consortium of world leaders in EDA, System on Chip (SoC) and System in Package (SiP) design such as Synopsys and STMicroelectronics gives the partners access to a large database of existing designs on which data-driven ML algorithms can be trained. This allows to overcome one of the main limitations of current academic efforts in using AI and ML for back-end, i.e., the very limited availability of training data.

Given that the main goal of the AMBEATion project is the development of a new EDA flow for analog layout, we believe that it fully aligns with the scope of the DATE 2024 conference. With respect to the Call for Paper, the project relates perfectly to tracks DT4 ("Design and Test for Analog and Mixed-Signal Circuits and Systems, and MEMS") and D14 ("Physical Analysis and Design").

The remainder of this paper is organized as follows. In Section II we illustrate the AMBEATion flow concept, providing a general overview of all the steps and abstraction levels involved. Then, Section III, provides an overview on the current implementation of each step. Perspectives, future research directions and lessons learned from the past months of the project are discussed in Section IV.

II. The AMBEATion Concept

At the highest level the AMBEATion Back-end flow focuses on the IC AMS placement phase. Therefore, the main element of the flow consists in a placer tool together with a set of supporting tools for realizing auxiliary functions which will be briefly described in the following section. The flow implements a novel mix of *top-down* placement steps, inspired by the techniques used in digital design, and *bottom-up* ones, specific of the analog domain [6].

Specifically, as illustrated in Figure 1, the AMBEATion flow is divided into three main sections: the first part processes the IC schematic, and is highlighted in pink; the second, generates the corresponding layout (green); finally, an offline phase is also included (blue) to train ML algorithms on a database of pre-existing designs. The schematic and layout processing sections are further split into five "levels", associated with an input-to-output flow with increasing abstraction levels:

a) The Input Level: deals with the processing of input schematic/netlist and layout database files. Layouts can also be an input to the flow during the training phase of ML algorithms, whereas they are only an output during normal

usage. The flow supports several industrial standards in terms of file format for schematic and layouts, including the IC CAD Open Access (OA) database format, as well as Circuit Design Language (CDL) netlists, and GDS-II layouts.

b) Level 0: contains simple pre-processing scripts aimed at simplifying the subsequent steps of the flow. In particular, it aims to recognize all classes of IC components which are present in the schematic and to differentiate devices based on different voltage classes or bulk connections. This level adds a layer of abstraction that makes the flow at least partially independent from the considered IC technology and device details. After Level 0, the rest of the flow is entirely realized with internal technology-independent netlist and layout database formats, based on JSON.

c) Level 1: is in charge of the generation of individual devices by processing PCells (or similar), and of their placement onto small topologies (such as current mirrors, differential pairs, capacitor dividers, etc). In order to do so, at the netlist level, a *Topology Recognition* script identifies different elementary functional topologies in the input schematic/netlist. At layout level, a *Device Placer* positions the devices' layouts generated by PCells within each identified topology, by respecting both general technology constraints and additional topology-specific constraints.

d) Level 2: deals with the placement of devices and elementary topologies identified at Level 1 onto an IP or block (e.g., operational amplifier, current/voltage reference, oscillator, etc). At netlist level, the *IP Recognition* block identifies the parts of the netlist belonging to each IP. At layout level, the *Group Placer* positions them in the layout. This placement process is closely interconnected with the device placement of Level 1, and multiple feedback iterations between the two are foreseen.

e) The Output Level: it's where the final placement layout is produced, as well as a new version of the netlist, annotated with parasitics, dummies, etc, and detailed reports on the flow execution.

The floorplanning of multiple IPs in the top-level circuit, considering signal/supply domains, clock routing for the digital part, etc, as well as packaging and bonding issues, are out of the scope of the automation flow foreseen in the AMBEATion project. However, the flow can be conceptually extended with additional abstraction levels (Level 3, 4, etc) to deal with these aspects in the future. As mentioned, the flow does not simply go from one level to the next in a "linear" way. Rather, the levels provide feedback to each other in an iterative way (e.g., the group placement will depend on the individual topologies placement, but will also influence the latter).

The offline training phase processes databases of human-generated inputs and outputs for any step that needs to be implemented by means of ML techniques, be it related to digital or analog components, and to any abstraction level. The obtained trained models are then used in inference-mode in various schematic- or layout-processing steps of the flow. In particular, Figure 1 is annotated with a "brain" icon to identify steps that internally use ML, and with a "wheel" icon

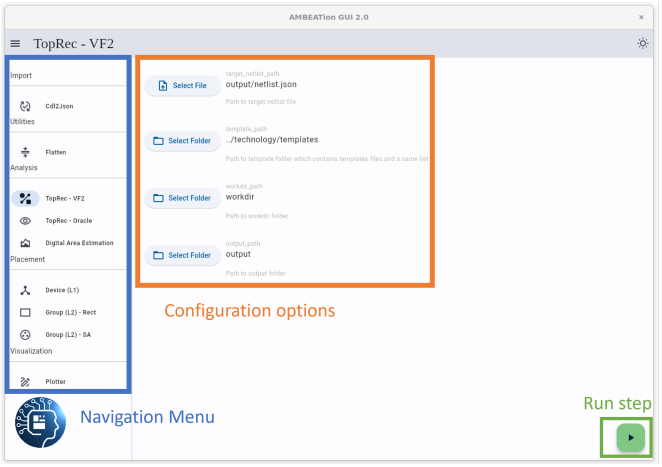

Fig. 2. An overview of the AMBEAtion GUI with main components highlighted. The navigation pane contains an entry for each pass of the AMBEATion flow currently implemented. Clicking on each item will display a different set of configuration options in the main window, which can then be changed by users. It also permits opening plots, log files, etc.

to indicate classic rule-based algorithms. As can be seen, for many steps, both classic and ML-based solutions are available, enabling a precise comparative assessment of the benefits of AI in this domain. Lastly, a "hand" icon indicates human-generated inputs. In the next section, we go in more detail on the flow steps implemented at the current state of the project.

III. Implementation

Similar to industrial EDA tools, the AMBEATion flow can be accessed both with a Command Line Interface (CLI) and with a simple Graphical User Interface (GUI), shown in Figure 2. The flow is built as an interconnection of modular software components, to simplify extensibility and maintainability, and to permit an easy swap between multiple implementations of the same step (e.g., classical vs ML-based, for comparison). To this end, all exchange of information between steps occurs through a unified file format based on the JSON standard. At Input Level, the JSON is substantially a translation of a common netlist format (CDL, OA, etc). Then, each step modifies or enriches this internal database, e.g., adding physical device information, placement coordinates, etc. The entire AMBEATion codebase is written in Python. In the following we briefly illustrate the functionality of each component of the flow.

A. Level 0 (Schematic): Input Conversion and Pre-processing

To implement Level 0 of Figure 1, a set of scripts have been developed to convert standard netlist formats into our internal JSON-based, vendor-independent "lingua franca", and assign each device to its respective class and voltage group, thus reducing the dependency on a specific technology node.

The main entities described in the internal database at this stage include: a) *cells*, i.e., sub-circuit definitions; b) *instances* of elementary devices or other sub-circuits within a cell; c) *libraries*, i.e., sets of cells, usually grouped by common

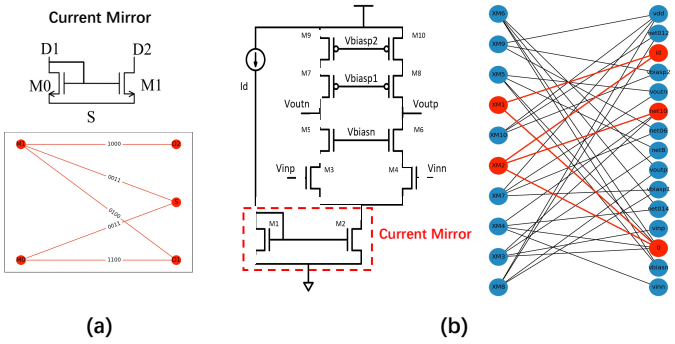

Fig. 3. Topology recognition. An NMOS current mirror primitive (a) is represented by a pattern graph, where vertices on the left side represent devices M0 and M1 respectively and those on the right side represent nets D1, D2 and S. Similarly, a fully differential Telescopic operational transconductance amplifier (OTA) (b) is converted into a bipartite graph and one subgraph is matched the current mirror pattern graph, highlighted by red.

functionality. Each entity contains several attributes, depending on its type. In later stages, the database will be enhanced with new entities, such as *topologies*, i.e., groups of devices identified at Level 1, forming a higher-level structure (e.g., a current mirror). Further, existing entities will receive new attributes, e.g., x/y layout coordinates for each device.

B. Level 1 (Schematic): Topology Recognition

The objective of this step is to identify functional topologies within the schematic, e.g., current mirrors, differential pairs, etc. This identification process is essential for accurately specifying constraints in the layout placement phase. Topology recognition can be converted to a subgraph isomorphism problem between a template graph (e.g. a current mirror) and the target graph (e.g., an operational amplifier). Namely, following [10], we adopt a bipartite graph representation for netlists, with two sets of nodes, one for devices and one for nets, as shown in Figure 3. The AMBEATion flow currently supports two topology recognition implementations: a classical one, based on the VF2 matching algorithm [11] and a ML-based alternative using Relational Graph Convolutional Neural Networks (RGCNs) [10], [12].

The main limitation of the VF2 implementation lies in its complexity, implying very long running time on large flattened netlists. Therefore, the orders of magnitude faster RGCN version can provide significant benefits in terms of efficiency for real world use-cases. On the other hand, the ML-approach can lead to significant accuracy degradations, which can be partially coped with by means of pre- and post-processing steps, as detailed in Section IV. The output file produced by topology recognition is a JSON database containing the original netlist enriched with the extracted topologies, with their associated type, device group, and nets.

C. Level 1 (Layout): PCell Processing and Device Placement

For what concerns the layout processing, Level 2 comprises two main steps: *PCell processing* and *device placement*. The former reconstructs the polygons comprising the device's layers, extracting the correspondence between spatial coordinates in the schematic and layout representations. Additionally, it can incorporate technology-specific constraints, such as spacing requirements, into the design. Currently, AMBEATion supports PCells [13], but processing of Python-based PyCells [14] and SKILL is also planned.

The Device Placer then elaborates the internal layout of each topology identified in the corresponding recognition step, using the geometrical information extracted from PCells, as well as technology-dependent constraints from a tech file. Furthermore, this step processes additional inputs relative to the type of dummy insertion required, and to the desired aspect ratio for each topology. The script then positions devices in matrix arrays which respect the user defined constraints. Moreover, to suppress the layout dependent effects (such as WPE, STI-stress etc) it reserves the necessary extra area around the matrix array. The optimization is purely analytical, and aims at approximating as well as possible the target aspect ratio, positioning devices in the appropriate number of rows.

The output of the script is an updated JSON netlist, where each device is annotated with the minimum (x_{min}, y_{min}) and maximum (x_{max}, y_{max}) coordinates of its bounding box within the topology it belongs to (if any). Coordinates are referred to (0,0) for each topology since this module does not deal with inter-topology placement, which is handled by the Level 2 Group Placer. The Device Placer is designed so that it can be iteratively invoked from within the optimization loop of the higher-level Group Placer, each time with different input parameters (e.g., different target aspect ratios). This gives the Group Placer the freedom to "re-shape" some topologies' bounding boxes to better fit the overall IP layout.

Importantly, digital blocks within an AMS IC are treated separately for what concerns device placement. Namely, they are not placed by the AMBEATion flow directly, but rather, after being identified by topology recognition, they are offloaded to a state-of-the-art digital EDA Place And Route (PnR) tool, given their high level of automation and QoR. Therefore, the Level 2 Group Placer will treat each digital block as a black-box, simply based on its layout area occupation. However, the placer should have the freedom to reshape/resize digital blocks, similarly to the case of analog topologies. To allow this, while avoiding time-consuming iterations including full digital PnR executions, the AMBEATion flow includes a fast ML-based *PnR Fesibility estimator* for digital logic.

1) Digital PnR Feasibility Estimation: The role of this component is to assess whether a digital block within an AMS design can be accommodated effectively within the designated layout area, without the need of a full PnR run.

In the current version of the AMBEATion flow, the Digital PnR Feasibility module has been implemented with a Decision Tree (DT) ML model, trained to assign a feasibility score to a design, based on high-level netlist and (expected) layout characteristics, as well as technology information. A DT was selected due to its simplicity, explainability, and effectiveness with limited training samples [15]. The model is trained in a

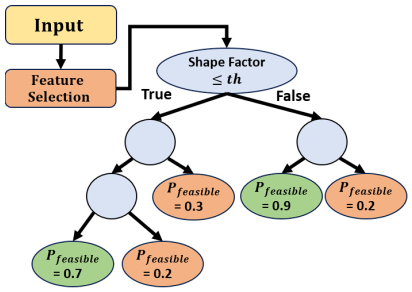

Fig. 4. Digital Area Estimation flow overview, consisting of a first feature selection step followed by the Decision Tree inference. The output of the flow is a probability of successful PnR.

supervised way, given examples of both successful and failed back-end executions of past designs. Its inputs are features that influence the chances of successfully closing a PnR, based on domain knowledge, e.g, the layout area hypothesis and its shape factor, the initial row utilization, the number of sequential cells in the design, the clock frequency, and the total number and density of pins. Technology information, such as the number of available routing layers is also provided. Once all features are available, a feature selection step based on cross-validation can then be applied to select the subset that best generalizes on unseen data.

After training, the model learns simple if-then-else rules that assign each new design to a "bin" (leaf node) by recursively comparing its features with learned thresholds. The leaf node value is a feasibility score (in [0:1]), as shown in Figure 4.

D. Level 2 (Schematic): IP Recognition

The goal of IP Recognition is to identify higher level structures in the circuit, such as operational amplifiers, oscillators, etc, and then constrain them accordingly for placement. Currently, this stage is not implemented, but techniques similar to those applied for Topology Recognition can be used for it too. The input, in this case, will include both individual devices and topologies identified at Level 1. ML-based approaches leveraging Graph Neural Networks are expected to work effectively in this case too, as demonstrated in [10].

E. Level 2 (Layout): Group Placer

The group placer (Level 2) places topology rectangles prepared by the device placer relative to each other. Thus, it forms the complete layout of a single IP block. Two alternative implementations are supported, leveraging respectively Simulated Annealing (SA) [16], [17], and Non-Linear Programming (NLP) [18], [19], two of the optimization methods considered state-of-the-art for this problem.

The SA placer leverages a "Sequence Pair" representation of the layout [20], which identifies virtual non-crossing pathways in the floorplan. Conversely, the NLP solution simply represents the x and y coordinates of the bottom-left corner of each group of devices as floating-point vectors, whose values are to be determined in the optimization process using gradient descent with adaptive estimation of first- and second-order moments [21].

Both placers can support the same optimization objectives, including area minimization, routability (e.g., Half-Perimeter Wire-Length or HWPL), etc. However, they are quite sensitive to their input hyper-parameters, as well as being entirely dependent on the L1 Placer results. To solve this, an agent-based approach is considered to optimize the placement of each IP. Namely, the group placement process can be controlled by an AI agent (e.g., a neural network trained with Reinforcement Learning, or an Evolutionary Algorithm), which will internally invoke the Device Placers with different input parameters (e.g. aspect ratio targets), and change the hyper-parameters of the SA and NLP optimizers, until all constraints are respected, and a combined objective function is minimized.

At the end, the output layout for each IP is exported both in the internal JSON database format, as well as in the standard GDS-II format. Other Output Layer adapter scripts allow the import of the generated layout into industrial EDA tools, to implement the following design phases (routing, signoff, etc).

F. Utilities

Besides the main scripts to implement key placement operations, the AMBEATion flow also includes several utilities, including basic scripts such as netlist flattening using the internal JSON format, as well as an interactive layout plotter, and several other import/export and conversion tools.

IV. Conclusions and Future Outlooks

Being a MSCA-RISE action, the main goal of the AMBEATion project is to put in contact European academic staff with leading industries, and vice versa, through the mechanism of secondments. In parallel, the overarching technical objective is to strengthen the competencies on AI-based AMS placement in Europe. To this end, the project takes strong inspiration from other non-EU initiatives, mainly from the U.S. NSF and DARPA programs [3], [4], building upon their work, and trying to further improve it. At the current stage, the main result achieved by the project is that of having built a flexible, modular AMS placement framework, which can function as foundational infrastructure for future research. Not only in the direction of developing new ML-enhanced EDA algorithms, but also in quantitatively and thoroughly comparing existing solutions, and in extensively evaluating them on *real-world* industrial use cases. To this end, the AMBEATion platform has been designed around a rich and easy-to-use language such as Python, and using an open, extensible and human-readable JSON-based database format. The flow is built for usability and extensibility, and is entirely cross platform. While the main evaluations have been performed on BCD8 technology by ST Microelectronics, the platform is also designed with technology independence in mind.

The most interesting results obtained up to the current stage concern the "real-world effectiveness" of ML-based AMS placement techniques. In fact, preliminary validation experiments on the flow revealed interesting insights, and

highlighted key limitations. We report here three of the main issues that have been identified during the project journey. Addressing these issues will also be the main direction of our future research in the remaining years of the project.

a) Data Quality and Variety: while the industrial partners involved in AMBEATion have access to large databases of past designs, the main obstacle towards obtaining highly accurate with ML-based placement approaches is data *quality* rather than quantity. The available datasets are often highly imbalanced, and contain many similar samples, that do not cover the design space entirely. For instance, input graphs used to train our RGCN models for topology recognition, despite including hundreds of thousands of nodes, contain very few examples of "uncommon" topologies (all structures except current mirrors are present in far less than 1% of the total nodes). Despite applying several countermeasures at training level, such as loss function weighting, oversampling, or treating the problem as a one-class anomaly detection, this imbalance still negatively affects performance in many cases. Similarly, the training data used for Digital PnR Feasibility Estimation contains many more examples of *successful* runs than failed ones, because the outputs and logs for the latter are usually not preserved by designers. One solution we envision and we explore in the future, is the use of *synthetic data generation* by means of scripts that create from scratch or modify existing netlists and layouts. This approach has proven effective in several other domains [22]. More broadly, these limitations also provides guidance on possible updates company policies on data storage and management.

b) Technology Independence: Many previous efforts on ML-based AMS layout focused on single technology nodes [3], [4], [10]. In AMBEATion, we found that one of the key issues in applying techniques from the state-of-the-art to build a general tool is that ML-based approaches, and often also classic ones, do not generalize well across technologies. As an example, the template library used by the VF2 algorithm for topology recognition had to be completely rewritten with respect to previous works [10] to make it work on BCD8, and new post-processing rules had to be designed to avoid false positive matches. Similar issues apply to ML models trained on one technology and applied to a different one. Similar to other domains, possible directions to address this issue for ML-based components are domain adaptation and transfer learning [23].

c) Scalability: The main practical limitation of many of the algorithms implemented in the AMBEATion flow is their limited scalability to large designs with hundreds of thousands of devices. Scalability has been addressed in digital design for years, and many of the same solutions (hierarchical placement, parallelization, etc) can be applied on AMS too. However, there are also some peculiarities. For example, the VF2 algorithm for topology recognition scales extremely well if it is applied independently, and possibly in parallel, to hierarchical blocks of the netlist (most of which correspond to small bipartite graphs). Conversely, it scales poorly on large flattened netlists. However, the hierarchical approach cannot identify topologies spread across multiple sub-circuit blocks, thus it requires designers to fully trust the goodness of their human-made hierarchy. The RGCN approach, conversely, executes fast even on flat netlists. In this sense, here and in many other parts of the flow, ML can be seen more as a way to improve performance by means of computational approximation, rather than a way to solve otherwise unsolvable problems.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101007730.

REFERENCES

[1] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1994.
[2] E. Malavasi *et al.*, "Automation of ic layout with analog constraints," *IEEE TCAD*, vol. 15, no. 8, pp. 923–942, 1996.
[3] K. Kunal *et al.*, "ALIGN: Open-Source Analog Layout Automation from the Ground Up," in *56th DAC*, 2019.
[4] B. Xu *et al.*, "MAGICAL: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence: Invited Paper," in *2019 IEEE/ACM ICCAD*, 2019, pp. 1–8.
[5] M. P.-H. Lin *et al.*, "Recent research development and new challenges in analog layout synthesis," in *ASP-DAC*, 2016, pp. 617–622.
[6] J. Scheible and J. Lienig, "Automation of analog ic layout: Challenges and solutions," in *ISPD*, 2015, p. 33–40.
[7] A. Venkatachar, "Confluence of ai/ml with eda and software engineering," in *ISQED*, 2021, pp. 13–15.
[8] A. Mirhoseini *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, pp. 207 – 212, 2021.
[9] B. Nikolić, "Ml for analog design: Good progress, but more to do," in *ACM/IEEE MLCAD*, 2022, pp. 53–54.
[10] K. Kunal *et al.*, "GANA: Graph Convolutional Network Based Automated Netlist Annotation for Analog Circuits," in *DATE*, 2020, pp. 55–60.
[11] L. Cordella *et al.*, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE TPAMI*, vol. 26, no. 10, pp. 1367–1372, 2004.
[12] A. Teliti, "Graph neural networks for topology recognition of ams integrated circuits," Master thesis, Politecnico di Torino, 2023.
[13] R. Arora *et al.*, "Virtuoso express pcells for better interoperability and performance on oa," *CDNLive! India 2007*, 2007.
[14] "Synopsys "pycell studio"." [Online]. Available: http://www.synopsys.com/cgibin/pycellstudio/req1.cgi
[15] O. Z. Maimon and L. Rokach, *Data mining with decision trees: theory and applications*. World scientific, 2014, vol. 81.
[16] Q. Ma *et al.*, "Simultaneous handling of symmetry, common centroid, and general placement constraints," *IEEE TCAD*, vol. 30, no. 1, pp. 85–95, 2010.
[17] Y. Li *et al.*, "Exploring a machine learning approach to performance driven analog ic placement," in *ISVLSI*. IEEE, 2020, pp. 24–29.
[18] B. Xu *et al.*, "Device layer-aware analytical placement for analog circuits," in *ISPD*, 2019, pp. 19–26.
[19] Y. Lin *et al.*, "Are analytical techniques worthwhile for analog ic placement?" in *2022*. IEEE, 2022, pp. 154–159.
[20] J.-M. Hsu and Y.-W. Chang, "A reusable methodology for non-slicing floorplanning," in *ASPCAS*, vol. 1, 2004, pp. 165–168 vol.1.
[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.
[22] J. Tremblay *et al.*, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *CVPR*, June 2018.
[23] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," *arXiv:1812.11806*, 2018.