Accelerating Cell-Aware Model Generation Through Machine Learning

Pierre D'hondt, Aymen Ladhar, Patrick Girard, Arnaud Virazel

Accelerating Cell-Aware Model Generation Through Machine Learning

P. d'Hondt [1, 2] A. Ladhar [1] P. Girard [2] A. Virazel [2]

[1] STMicroelectronics
Crolles, France
pierre.dhondt@st.com, aymen.ladhar@st.com

[2] LIRMM – Univ. of Montpellier / CNRS
Montpellier, France
girard@lirmm.fr, virazel@lirmm.fr

I. Introduction

To achieve the highest product quality, Cell-Aware (CA) test has become mandatory for semiconductor industry. In this methodology, a cell-internal-fault dictionary or *CA model*, describing the detection conditions of each potential defect affecting a cell, is used [1-2]. However, the generation of CA models for all standard cells is a time- and resource-consuming task that limits the deployment of CA test.

Typical CA model generation flow starts with a SPICE netlist representation of a standard cell. This representation is used by an electrical simulator to simulate each potential defect against an exhaustive set of stimuli. The stimuli detecting defects are synthetized into a CA model. As thousands of standard cells, with various complexities, are used for a given technology, the generation time of CA models for complete standard cell libraries may reach up to several months, thus drastically increasing the library characterization process cost.

To improve the generation run time of CA models and ease the characterization, this work proposes a methodology to predict the behavior of cell-internal defects using Machine Learning (ML) [3]. More widely, the goal is to use existing CA models from various standard cell libraries developed using different technologies to predict CA models for new standard cells independently of the technology.

II. Learning-Based CA Defect Characterization

A. Data representation

The first challenging task is to describe cell transistor netlist as well as corresponding cell-internal defects in a uniform (standardized) ML-friendly manner, so that a ML algorithm can learn and infer from data irrespective of their incoming library and technology. In our work, a matrix representation of cells and corresponding defects was chosen to this purpose. Table I shows an example of matrix representation for an NAND2 cell. It is composed of four types of information:

- *Cell patterns and responses*. This gives the values applied on input (A, B) as well as the cell response on output Z. It uses a four-value logic algebra made of 0, 1, R (rising from 0 to 1) and F (falling from 1 to 0).
- *Cell description*. This indicates the active or inactive state of every cell transistor (referred to as Ni for nmos and Pi for pmos), when a given pattern is applied on cell inputs. Each transistor can be in the following state: active (1), passive (0), switching to active state (R), switching to passive state (F).
- *Defect description*. This part lists all transistor terminals as a column. To describe an open defect, a value '1' indicates the transistor terminal affected by the defect. For a short defect, a value '1' on two transistor terminals indicates that a short exists between these two terminals.
- *Defect detection*. This is the class (label) of the data sample (the output of ML classifier). A value '1' ('0') means that the defect is detected (undetected) by the input pattern.

TABLE I. Example of Matrix Representation for a NAND2 Cell

Cell inputs & responses			**Cell description**				**Defect description**				**About defect**		**Defect detection**
A	**B**	**Z**	**N0**	**N1**	**P0**	**…**	**N1_D**	**N1_G**	**N1_S**	**…**	**name**	**type**	**fZ**
0	0	1	0	0	1	…	0	0	0	…	free	free	0
0	1	1	0	1	1	…	0	0	0	…	D7	ope	0
0	F	1	0	F	1	…	0	0	0	…	free	free	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	1	1	0	1	1	…	1	0	1	…	D15	short	1
1	1	0	1	1	0	…	1	0	1	…	D15	short	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

B. Proposed Defect Characterization Methodology

The proposed learning-based defect characterization methodology is used to predict the behavior of a cell affected by intra-cell defects, hence avoiding costly electrical defect simulations. The proposed flow is sketched in Fig. 1. It is based on supervised learning that takes a known set of input data and known responses (*labeled data*) used as training data, trains a model to classify those data, and then uses this model to predict (*infer*) the class of new data. In this work, we used a Random Forest Classifier.

Training data are matrices containing the above four types of information. They are obtained from existing CA models, formerly generated by relying to brute-force electrical defect simulations. The ML algorithm is trained to predict the defect detection label, using the first three types of information.

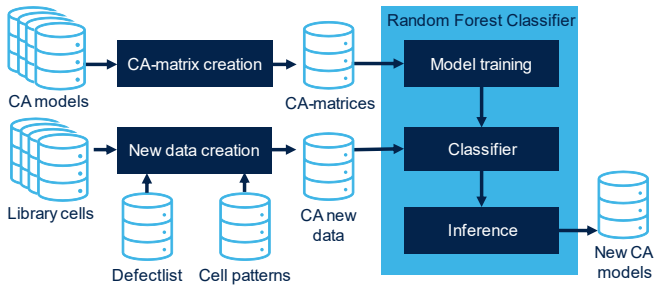

Fig. 1: Generic view of the ML-based defect characterization flow

New data represent the cells to be characterized and only contains the first three types of information. The class (label) of the new data instance is missing. The ML classifier is used to predict that class.

III. Experimental Results

We implemented our method in a python program. The ML algorithms were taken from the publicly available python module called scikit-learn. Our dataset was composed of 1712 standard cells coming from standard cell libraries developed using three technologies (C40 (446 cells), 28SOI (825 cells) and C28 (441 cells)). All these cells already had a CA model generated by a commercial tool using electrical simulations. The method was experimented in two different ways. First, the ML model was trained and evaluated using cells belonging to one technology. Second, we trained the model on one technology and evaluated it on another one.

A. *Predicting defect behavior on the same technology*

We first trained the ML model on cells of 28SOI standard cell libraries. For *m* cells available in a given group, we trained the ML model over *m-1* cells and evaluate its **prediction accuracy** on the *m*-th cell. A loop ensured that each cell is used as the *m*-th cell. On average, a group contains 8.6 cells. In this work, we considered all possible open and short defects (static and dynamic) in each cell. For open defects, the average prediction accuracy is 99.9%, with most of the cells at 100% accuracy. Results achieved for short defects are similar.

B. *Predicting defect behavior on another technology*

We also conducted experiments on cells belonging to two different technologies. Here, the ML model was trained over all available cells of a given technology and the evaluation was done on one cell of another technology. A loop was used to allow all cells of the second technology to be evaluated. First experiments consisted in training the ML model on 28SOI cells and evaluating on C28 cells. The average prediction accuracies are globally lower compared to those of previous part. After investigating on this point, we noticed that the behavior of most of the cells (68% of cells) is accurately predicted (accuracy > 97%), while accuracy for few cells is quite low. This phenomenon is discussed in section IV. Second experiments aimed at verifying the efficiency of our method when different transistor sizes are considered. We trained the ML model over the 28SOI standard cells and used it to predict the behavior of C40 cells. This time, 80% of cells were accurately predicted proving that our ML-based characterization methodology could be used to generate CA models for a (large) part of cells of a new technology.

IV. Discussion - Conclusion

We analyzed cells for which the defect characterization methodology gives excellent prediction accuracy as well as those for which the prediction accuracy was quite low. On one hand, we noticed that cells with good prediction have a cell with a similar schematic in the training group. On the other hand, cells leading to poor prediction accuracy may have (i) new logic functions that do not appear in the cells of the training dataset, or (ii) a transistor configuration which is completely new when compared to cells in the training dataset.

Considering the above analysis, it appears that the ML-based CA model generation flow cannot be used for all cells in a standard cell library. So, we propose the hybrid flow sketched in Fig. 2 to accelerate the CA model generation. When the CA model for a new cell is needed, we first check if the ML-based generation will lead to high-quality CA models. This is done by analyzing the schematic (structure) of the new cell and check whether the training dataset contains a similar cell. If the ML algorithm is expected to give good results, the new cell is prepared (representation in a matrix) and submitted to the trained ML algorithm. The output information is then parsed to the desired file format. Conversely, if the ML algorithm is expected to give poor prediction results, the standard simulation-based generation flow is used to obtain the CA model. A feedback loop uses this new simulated CA model to supplement the training datasets and improve the ML algorithm for further prediction.

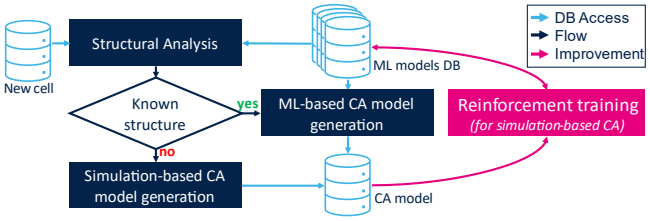

Fig. 2: Hybrid flow for CA model generation

While using 28SOI cells to obtain CA models for 409 cells in C40 technology, we found that 205 (50%) of them can go through the ML-based CA model generation. It requires 21947 seconds (~ 6 hours) to complete, using a single CPU core. Considering that a simulation-based generation for these 205 cells would require ~78 days, we can estimate the reduction in generation time to **99.7%**. For the 204 cells needing a simulation-based generation, the generation time was estimated to ~172 days (~ 5.7 months) considering a single SPICE license. Now, if we consider the C40 group composed of 409 cells, the hybrid flow represents a reduction in generation time of about **38%**.

After investigating results of these experiments, we observed that the ML-based CA model generation works well for about 80% of cells of the C40 group. Surprisingly, the structural analysis revealed that only 50% could be evaluated using the ML-based generation part of the flow. This shows that there is still room for further improvement of the structural analysis in our flow, and hence get better performance of the ML-based CA model generation process.

To conclude, experiments in sections III.A and III.B have been carried out on a reasonable size (1712) of standard cell population. Considering that more than 10000 cells have usually to be characterized for a given technology, the hybrid flow in Fig. 2 is expected to provide even better results, especially owing to the reinforcement training that uses simulation generated models for supplementing the training datasets, and hence reduce the number of electrical simulations.

References

[1] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, P. Weseloh, M. Wittke, M. Kassab, and C. W Schuermyer, "Cell-Aware Fault Model Creation And Pattern Generation," US Patent 12/718,799, 2010.

[2] S. Mhamdi, P. Girard, A. Virazel, A. Bosio and A. Ladhar, "A Learning-Based Cell-Aware Diagnosis Flow for Industrial Customer Returns," in *Proc. IEEE International Test Conf.*, 2020.

[3] P. d'Hondt, A. Ladhar, P. Girard, A. Virazel, "A Learning-Based Methodology for Accelerating Cell-Aware Model Generation" *in Proc. Design, Automation and Test in Europe Conference, 2021.*